UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Valerio Rosset

UM MODELO DE AUTORIZAÇÃO E DISTRIBUIÇÃO DE DIREITOS DE ACESSO SOBRE CONTEÚDOS DIGITAIS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Carla Merkle Westphall, Dra.
Professora Orientadora

Florianópolis, Fevereiro de 2004

Um Modelo de Autorização e Distribuição de Direitos de Acesso Sobre Conteúdos Digitais

Valerio Rosset

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Concentração em Sistemas de Computação e aprovada em sua forma final pelo programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Raul Sidnei Wazlawick, Dr.
Coordenador do programa de pós-graduação
em Ciência da Computação.

Banca Examinadora:

_____

Prof. Dra. Carla Merkle Westphall.
Orientadora

_____

Prof. Dr. Roberto Willrich.

_____

Prof. Dr. Mário Antonio Ribeiro Dantas.

_____

Prof. Dr. Joni da Silva Fraga.

“Aos que possuem conhecimento, o futuro!
Aos que possuem poder, o presente!
Aos que possuem a sabedoria,
a imortalidade !”
(Valerio Rosset).

Este trabalho é dedicado a todos que lutam por dias melhores,
permitindo que honra, lealdade, amizade
e o companheirismo ajudem a superar
as dificuldades impostas no
decorrer do caminho.

Agradeço a minha família, novamente pela confiança depositada.
A professora Carla, pelas orientações e ensinamentos.
Aos amigos que sempre estiveram presentes
durante a caminhada.
A Deus ...

Sumário

LISTA DE FIGURAS

LISTA DE SIGLAS

AADD	- Aplicação de Autorização e Distribuição de Direitos
ACL	- Access Control List
ACP	- Access Control Processor
B2B	- Business to Business
B2C	- Business to Consumer
DAC	- Discretionary Access Control
DES	- Data Encryption Standard
DOD	- Department of Defense
DOM	- Document Object Model
DRM	- Digital Rights Management
DSS	- Digital Signature Standard
DTD	- Document Type Definition
ECC	- Elliptic Curve Cryptography
HTML	- Hyper Text Markup Language
IETF	- The Internet Engineering Task Force
IDEA	- International Data Encryption Algorithm
ISO	- International Standard Organization
MAC	- Mandatory Access Control
OASIS	- Organization for the Advancement of Structured Information Standards
OSI	- Open Systems Interconnect
PAP	- Policy Administration Point
PDP	- Policy Decision Point
PEP	- Policy Enforcement Point
PIP	- Policy Information Point
PKI	- Public Key Infrastructure
PRP	- Policy Retrieval Point
RBAC	- Role Based Access Control
SAML	- Secure Assertion Markup Language
SAX	- Simple API for XML
SGML	- Structured General Markup Language
SHA	- Secure Hash Algorithm
SOAP	- Simple Object Access Protocol
SPKI	- Simple Public Key Infrastructure
TCB	- Trusted Computing Base
TI	- Tecnologia da Informação
UDDI	- Universal Description Discovery and Integration
URI	- Universal Resource Identifiers
URN	- Uniform Resource Names
XACML	- eXtensible Access Control Language
XML	- eXtensible Markup Language
XKMS	- XML Key Management Specification
XSLT	- Extensible Style Language Transforms
X-KISS	- XML Key Information Service Specification
X-KRSS	- XML Key Registration Service Specification
WSDL	- Web Services Description Language
W3C	- World Wide Web Consortium

RESUMO

Atualmente um crescente volume de informações está sendo distribuído através da *web*. Essas informações podem estar, por exemplo, na forma de *E-Books* (*Electronic Books*), impondo a necessidade de controlar o acesso e a distribuição desses conteúdos digitais, bem como o controle de uso desses conteúdos através da *web*. Esta dissertação propõe um modelo de autorização e distribuição de direitos de acesso, utilizando padrões de segurança XML para suporte e integração de aplicações que visam o controle de uso de conteúdos digitais, como em sistemas DRM (*Digital Rights Management*). O modelo está baseado em uma recomendação de arquitetura distribuída de autorização e controle de acesso da IETF chamada AAA *Authorization Framework*. O modelo de autorização e distribuição de direitos de acesso foi implementado através de uma aplicação denominada AADD (*Aplicação de Autorização e Distribuição de Direitos*), que utiliza os padrões de segurança SAML e XACML, para transportar informações sobre direitos de acesso e implementar políticas de controle de acesso.

Palavras-chave: Distribuição de Direitos de Acesso; Controle de Uso; XML; Segurança;

ABSTRACT

Nowadays an increasing amount of information is being distributed by the web. This information can be, for example, in Electronic Books form, imposing the necessity of to control access and distribution of these digital contents by the web. This dissertation presents a model of access rights authorization and distribution architecture using XML security standards for to support and integrate applications that enforce the usage control of digital contents, like Digital Rights Management systems. The model is based in an IETF recommendation of an access control and authorization distributed architecture (AAA Authorization Framework). The model was implemented such as an application named Rights Distribution and Authorization Application. This application uses the SAML and XACML standards to transport access rights information and implement access control policies.

Keywords: Access Rights Distribution; Usage Control; XML; Security;

1. INTRODUÇÃO

1.1. MOTIVAÇÃO

Muitos são os esforços para definir como controlar o acesso e a distribuição de conteúdos digitais, bem como o controle de uso desses conteúdos através da *web* [CHONG 2002], [KRAFT 2002] e [PARK 2002]. Porém, os trabalhos presentes na literatura não demonstram como efetuar a integração de tecnologias e padrões para o fortalecimento da segurança. Esta dissertação propõe um modelo de autorização e distribuição de direitos de acesso utilizando padrões de segurança XML para suporte e integração de aplicações que visam o controle de uso de conteúdos digitais, como os sistemas DRM (*Digital Rights Management*) [CHONG 2002] [FILIPPIN 2004].

As características de flexibilidade e interoperabilidade obtidas através do uso de padrões de segurança XML podem facilitar o gerenciamento de direitos digitais através da *web*. O uso de um padrão para escrita de políticas de controle de acesso em XML, por exemplo XACML [GODIK 2003], pode proporcionar alto poder de flexibilidade na definição de regras de autorização, que permitem desde a definição de políticas globais genéricas até a definição de políticas totalmente individuais. Outra vantagem é a capacidade de interoperabilidade com aplicações que possam estar em domínios diferentes, proporcionando a capacidade de executar controle de acesso distribuído. A interoperabilidade também ocorre com a troca de mensagens entre as entidades de um sistema, o uso de padrões de troca de mensagens seguras, como SAML [MALLER 2003], permitem que entidades comuniquem-se umas com as outras independentemente do ambiente em que se encontrem.

1.2. OBJETIVOS

A distribuição controlada sobre conteúdos digitais é efetuada através da aplicação da idéia de controle de uso de conteúdos definido em um contexto DRM [CHONG 2002]. Geralmente, o controle de uso é realizado com base em licenças de uso. Uma licença de uso carrega em seu conteúdo informações de direitos de um usuário sobre um

conteúdo. Os direitos presentes em uma licença de uso são definidos através de um processo de autorização, que ocorre com base na avaliação de políticas de controle de acesso, definidas para um conteúdo específico. Em um contexto DRM, uma entidade responsável por definir licenças de uso não necessariamente precisa executar o processo de autorização, esse processo pode ser executado por uma entidade exclusivamente designada para esse fim.

Assim o objetivo desse trabalho é definir e testar um modelo autorização e distribuição de direitos de acesso sobre conteúdos digitais. O modelo determina como ocorre o processo de avaliação de políticas de controle de acesso para disponibilizar direitos de acesso para confecção de licenças de uso para conteúdos digitais protegidos. Também apresenta uma visão geral de como os padrões de segurança XML podem interagir para complementar a segurança na troca de mensagens entre entidades no contexto DRM. O modelo ainda determina como o processo de autorização pode ser aplicado a requisições de acesso. Finalmente, é implementada uma aplicação para a avaliação de políticas de autorização e distribuição de direitos de acesso.

1.3. JUSTIFICATIVAS

A necessidade de controlar o uso de conteúdos digitais, para evitar fraudes e violação de direitos autorais, é a principal justificativa para se realizar pesquisas no sentido de definir tecnologias capazes de realizar esse controle, sem ferir direitos adquiridos de consumidores sobre conteúdos protegidos.

Contudo, uma arquitetura DRM deve possuir diferentes entidades responsáveis por desempenhar diferentes papéis, como geração de licenças de uso e direitos, sendo que a geração de uma licença de uso depende diretamente do fornecimento de direitos de um sujeito. Assim pode-se determinar a necessidade de definir uma base de autorização que determine que tipos de direitos sujeitos possuem sobre objetos. Finalmente, em um ambiente distribuído, ocorre a necessidade de garantir a segurança na troca de informações que disponibilizam direitos.

1.4. ESTRUTURA DA DISSERTAÇÃO

Este trabalho está organizado da seguinte forma: no capítulo 2 são apresentados conceitos básicos de segurança computacional. No capítulo 3 são apresentados conceitos sobre *Web Services*, XML e os padrões abertos de segurança XML, definindo suas estruturas e características. No capítulo 4 são apresentados os trabalhos relacionados. No capítulo 5 é apresentado o modelo de autorização e distribuição de direitos de acesso seguido pela implementação apresentada no capítulo 6. Por fim, no capítulo 7 são apresentados as conclusões e os trabalhos futuros.

2. CONCEITOS BÁSICOS DE SEGURANÇA COMPUTACIONAL

A segurança trata essencialmente da proteção de recursos [GOLLMANN 1999]. Isso implica que se deve conhecer os recursos e seus valores pra melhor protege-los. Prevenção, detecção e reação são os princípios da segurança computacional. Exemplos de segurança física no mundo real podem ajudar a esclarecer esses princípios de segurança.

Quando alguém tranca as portas e as janelas de sua casa. Pode se dizer que ela está se prevenindo contra uma possível ação de um assaltante. Ainda assim este mesmo assaltante pode entrar por uma porta ou janela acidentalmente aberta ou ainda pela chaminé. Sendo assim o assaltante pratica o assalto. A invasão só pode ser detectada quando você retorna e encontra as pegadas do assaltante ou não encontra seus pertences, infelizmente a detecção do assalto foi tardia, mas isso poderia ser resolvido com um sistema de alarme. Depois de descobrir que foi roubado é necessário tomar uma reação, nesse caso a melhor seria chamar a policia.

Geralmente as ações não autorizadas são frutos de falhas de segurança previstas, como, por exemplo, deixar portas e janelas destrancadas, ou ainda por falhas de segurança não previstas, como o ladrão entrar pela chaminé. Nos sistemas computacionais as falhas de segurança seguem o mesmo principio.

Uma analogia pode ser feita no sentido de que em sistemas computacionais as portas de comunicação de rede podem representar um perigo quando estiverem totalmente abertas, para usuários não autorizados. Esse é o tipo de falha de segurança acidental ou de configuração. Seguindo a analogia uma “chaminé”, em um sistema computacional, representa alguma falha de segurança desconhecida existente no sistema, que pode ser explorada por um usuário não autorizado. Esse é o tipo de falha de segurança não prevista, a qual consiste no termo vulnerabilidade de um sistema.

2.1. SEGURANÇA COMPUTACIONAL

A segurança computacional pode ser definida como prevenção e detecção de ações não autorizadas em um sistema computacional. A segurança computacional prevê o cumprimento das seguintes propriedades fundamentais [STALLINGS 1999]: a confidencialidade**:** prevenção contra a revelação não autorizada de informação; a Integridade**:** prevenção contra a alteração não autorizada de uma informação; e a Disponibilidade**:** prevenção contra a retenção não autorizada de uma informação ou recurso.

Esses aspectos não são únicos e suficientes para definir o nível de comprometimento de um recurso ou informação. Outros aspectos também podem ser abordados como, por exemplo, a autenticação e responsabilidade.

A definição desses aspectos mostra como a prevenção e a detecção de ações não autorizadas podem ser atingidas. Porém, alem dos aspectos básicos da segurança computacional, é também de grande importância a inclusão de conceitos de mecanismos de segurança como autorização e controle de acesso. A autorização e o controle de acesso são aplicados baseados em políticas de segurança.

2.1.1. Confidencialidade

A confidencialidade está intimamente ligada aos termos de segredo e privacidade. Estes termos são usualmente utilizados para distinguir a proteção entre dados pessoais (privacidade) e dados pertencentes a uma organização (segredo) [STALLINGS 1999]. A confidencialidade foi um aspecto muito pesquisado no campo da segurança computacional e possui um conceito bem definido. Pode-se dizer que garantir a confidencialidade de uma informação é tornar possível que apenas pessoas autorizadas possam conhece-la. Algumas vezes, segurança e confidencialidade são utilizadas erroneamente como sinônimos.

2.1.2. Integridade

A Integridade está ligada a certeza de que uma informação realmente possui um conteúdo verdadeiro e original, mesmo que esta informação esteja transitando em um sistema computacional. Por exemplo, em uma comunicação entre sistemas computacionais a informação que sai de uma origem deve ser igual a informação que chega ao destino. Em [CLARK e WILSON 1987] a integridade é uma propriedade que

determina que a nenhum usuário de um sistema, mesmo que autorizado, pode modificar elementos de dados de modo que recursos ou registros contábeis de uma companhia sejam perdidos ou corrompidos.

2.1.3. Disponibilidade

A disponibilidade trata da alocação de recursos e informações sempre que sejam necessários. O Instituto de Padrões Internacional ISO 7498-2 [ISO 1988] na ISO/OSI arquitetura de segurança para comunicação segura, define disponibilidade como a propriedade para tornar um recurso acessível e utilizável para entidades autorizadas.

Na segurança computacional garantir a disponibilidade de um recurso é prevenir contra a possibilidade desse recurso se tornar indisponível para o acesso de usuários legítimos. Quando um recurso fica bloqueado é caracterizada uma negação de serviço.

2.1.4. Autenticação

A autenticação trata da certificação de que a origem de uma informação é identificada corretamente, e que a identidade não é falsa. Autenticação assegura que uma comunicação é autentica [STALLINGS 1999].

2.1.5. Responsabilidade

O principio da responsabilidade determina que informações de auditoria precisam ser mantidas e protegidas para que ações que possam afetar o sistema sejam rastreadas para encontrar a parte responsável [GOLLMANN 1999].

2.2. AMEAÇAS DE SEGURANÇA, ATAQUES E VULNERABILIDADES

Uma ameaça pode ser caracterizada como uma inesperada e potencial ação, que pode geralmente produzir danos com proporções não desejáveis a um sistema computacional. Edward Amoroso [AMOROSO 1994] define ameaça como alguma coisa má que pode acontecer.

As ações inesperadas que definem uma ameaça podem ter uma origem intencional ou não intencional. Ações não intencionais não são premeditadas e podem ser caracterizadas como falhas de software e hardware. Ações intencionais são

premeditadas e geralmente são realizadas com base em algum conhecimento de vulnerabilidades de sistemas computacionais.

A vulnerabilidade de um sistema consiste na existência de uma ameaça. Amoroso [AMOROSO 1994] define vulnerabilidade como alguma característica inoportuna que torne possível a ocorrência de uma ameaça em potencial. Em outras palavras a presença de uma vulnerabilidade permite que danos possam acontecer em um sistema computacional.

A exploração de uma vulnerabilidade de um sistema computacional, por parte de uma ação intencional, consiste em um ataque ao sistema. "Um ataque a um sistema computacional é qualquer ação executada por um intruso malicioso que envolve a exploração de uma vulnerabilidade causando a ocorrência de uma ameaça existente" [AMOROSO 1994].

O papel fundamental da segurança computacional é prover técnicas e metodologias para minimizar os ataques, garantindo o cumprimento das propriedades fundamentais.

2.2.1. Tipos de Ameaças

As ameaças de segurança podem ser divididas em três grupos: ameaças de revelação, ameaças de integridade e ameaças de negação de serviço [GOLLMANN 1999].

A ameaça de revelação envolve a disseminação da informação para um individuo para quem esta informação não poderia ser vista. No contexto de segurança computacional, esse tipo de ameaça ocorre sempre que algo secreto, que está armazenado em um sistema computacional ou trafegando entre sistemas computacionais, é revelada a alguém que não poderia conhecer o segredo.

A ameaça de integridade envolve uma alteração não autorizada da informação que está armazenada em um sistema computacional ou quando está transitando entre os sistemas computacionais. Quando um intruso malicioso altera alguma informação, considera-se que a integridade da informação foi comprometida. A integridade também é comprometida quando a informação é alterada de maneira inocente, através de um erro, provocando uma alteração não autorizada.

A ameaça de negação de serviço ocorre sempre que o acesso a algum sistema computacional é intencionalmente bloqueado como resultado de uma ação maliciosa

executada por um outro usuário. Quando um usuário requer acesso a um serviço e outro usuário faz algo malicioso para evitar o acesso, é caracterizada uma negação de serviço.

2.2.2. Tipos de Ataques

Um ataque é a concretização de uma ameaça. Conseqüentemente cada tipo de ataque vai pertencer a um tipo de ameaça. Segundo [SOARES 1999] os principais ataques possíveis de ocorrer um sistema computacional, são os seguintes: Personificação: ocorre quando uma entidade faz-se passar por outra; Replay: ocorre quando uma mensagem, ou parte dela é interceptada, e posteriormente transmitida pra produzir um efeito não autorizado; Modificação: o conteúdo de uma mensagem é alterado implicando em efeitos não autorizados sem que o sistema consiga detectar a alteração; Negação de serviço: ocorre quando uma entidade não executa sua função apropriadamente e não permite que outras entidades executem suas funções; Ataques internos: ocorrem quando usuários legítimos comportam-se de maneira não autorizada ou não esperada; Armadilhas (*TrapDoor*): ocorre quando uma entidade do sistema é modificada para produzir efeitos não autorizados em resposta a um comando ou evento premeditados; Cavalos de Tróia: ocorre quando uma entidade executa funções não autorizadas, em adição às que está autorizada a executar.

2.3. POLÍTICAS DE SEGURANÇA

2.3.1. Definição

O termo de política de segurança denota sobre o aspecto organizacional estabelecendo como um sistema deve proceder para difundir ou armazenar informações de maneira segura. A política de segurança de um sistema computacional é definida como um conjunto de regras e praticas que determinam a maneira pela qual as informações e recursos são gestionados, protegidos e distribuídos no interior de um sistema específico [ISO/IEC 15408-1, 1999].

Na literatura as políticas de segurança são divididas em políticas de segurança física e políticas de segurança lógica [WESTPHALL 2000].

As políticas de segurança física determinam como o sistema deve ser protegido fisicamente. Já as políticas de segurança lógica tratam da segurança das informações

que estão armazenadas ou sendo trocadas em um sistema computacional, através de ferramentas e controles internos.

Políticas de segurança lógica podem ser divididas em várias partes. Para identificar e autenticar usuários de um sistema computacional são necessárias a utilização de políticas de identificação e autenticação. Depois de identificado o usuário o sistema deve definir quais ações são autorizadas ao usuário em particular, através do uso de políticas de autorização ou políticas de controle de acesso.

2.3.2. Políticas de Autorização e Controle de Acesso

Políticas de autorização e controle de acesso podem ser classificados em políticas discricionárias, políticas obrigatórias ou não-discricionárias e políticas baseadas em papéis.

As políticas discricionárias determinam o acesso a informação baseada na identidade do usuário e regras que especificam, para cada usuário (ou grupo de usuários) e para cada objeto no sistema, os modos de acesso específicos que o usuário possuí sobre um objeto (e.g., Leitura, escrita ou execução) [SANDHU e SAMARATI 1994].

Cada requisição de acesso a um objeto é precedida da verificação da autorização específica. Se a autorização existe e afirma que o usuário pode acessar um objeto no modo específico, é permitido o acesso, senão o acesso é negado. As políticas discricionárias podem ser utilizadas em uma grande variedade de sistemas pelo fato de possuírem flexibilidade. Por outro lado, as políticas discricionárias podem não promover uma real segurança quanto ao fluxo de informações em um sistema, pois é possível ultrapassar restrições de acesso estabelecidas pelas regras de autorização. Por exemplo, um usuário que possui acesso a uma determinada informação pode transferi-la livremente a outros usuários que não poderiam ter acesso à suposta informação. As políticas discricionárias não controlam a disseminação de informação.

As políticas obrigatórias determinam o acesso a informação baseada na classificação de sujeitos e objetos em um sistema. Cada usuário e cada objeto pertencem a um nível de segurança [SANDHU e SAMARATI 1994]. O nível de segurança a que um objeto pertence determina qual a sensibilidade ou o grau de segurança de uma informação contida nele. O nível de segurança associado a um usuário, chamado de habilitação (*clearance*), determinada que o usuário não distribua informação a usuários

que não sejam autorizados a vê-la. A classificação de sujeitos e objetos é feita através do uso de etiquetas para cada um deles. Assim os objetos de um sistema são etiquetados de acordo com sua classificação específica e nível de segurança a qual pertencem. Já o sujeito é etiquetado com o nível de segurança ou habilitação a que tem acesso. A autorização do acesso será determinada comparando a habilitação de um usuário com a classificação do objeto. O mecanismo utilizado para etiquetar sujeitos e objetos de acordo com a classificação e habilitação é chamado de rótulo de segurança.

Políticas baseadas em papéis determinam o acesso de um usuário a uma informação com base nas atividades que cada usuário executa em um sistema computacional, ou seja, é baseado na função ou papel de cada usuário. Um papel pode ser definido como um conjunto de ações e responsabilidades associadas a uma atividade de trabalho em particular [SANDHU e SAMARATI 1994]. Assim, o controle de acesso aos objetos é especificado por papéis. Um usuário que possua uma determinada função tem permissão para executar operações para as quais a função é autorizada. É possível determinar que vários usuários possuam a mesma função. Assim como os usuários são classificados por funções os objetos também podem ser classificados de acordo a sua função, por exemplo, vários objetos podem pertencer a uma única classe. Um usuário pode ter permissão para acessar não um objeto específico, mas também qualquer objeto que pertença à mesma classe.

2.3.3. Monitor de Referência

Virtualmente todo sistema computacional pode ser modelado em termos de sujeitos acessando objetos [AMOROSO 1994]. Um sujeito pode ser um usuário de um sistema computacional identificado por algum tipo de autenticação. O objeto é um recurso que o sujeito ou usuário tem acesso em determinado sistema computacional. Assim um sujeito precisa fazer uma requisição de acesso para um determinado objeto. Neste contexto pode-se imaginar que é necessária uma estrutura de decisão para determinar quais requisições podem ser permitidas e quais não podem ser. Essa estrutura responsável pelas decisões é chamada monitor de referência [LAMPSON 1971].

O monitor de referência tomará as decisões referentes a cada requisição de acesso baseado nas políticas de segurança definidas para os usuários de um sistema

computacional. Assim uma política de segurança de um sistema computacional define as condições e como os acessos serão intermediados por uma funcionalidade de um sistema monitor de referencia [AMOROSO 1994].

A funcionalidades de um monitor de referencia são implementadas através de mecanismos de segurança. O conjunto de recursos de hardware e software que implementam o conceito de monitor de referência é definido como núcleo de segurança [LANDWEHR 1983]. Assim pode-se afirmar que mecanismos de segurança são elementos necessários para a implementação das políticas de segurança.

2.3.4. Modelo de Segurança e Políticas de Segurança

A maneira como as políticas de segurança são escritas é de grande importância para a implementação dessas políticas em um sistema computacional. Um modelo de segurança especifica como as políticas devem ser escritas ou expressas em um determinado sistema computacional. Defini-se modelos de segurança como uma representação restrita de uma classe de sistemas que abstrai detalhes de modo a realçar uma propriedade específica ou um conjunto de comportamentos. Modelos de segurança são úteis para definição de políticas especificas de segurança [AMOROSO 1994].

2.4. MODELOS DE SEGURANÇA

O termo modelos de segurança também é chamado, na literatura, como modelos de controle de acesso. Um modelo de segurança determina a implementação das políticas de segurança para um sistema. Modelos de segurança são formas de descrever as políticas de autorização, determinando tanto o comportamento de entidades governadas pela política quanto às regras que definem a evolução desta política [GOLLMANN 1999].

Alguns modelos aplicam-se em ambientes onde as políticas são estáticas, como o BLP [BELL E LAPADULA 1976], e outros consideram mudanças dinâmicas dos direitos de acesso, como o modelo Chinese Wall [BREWER e NASH 1989]. Alguns modelos descrevem políticas para manter propriedades como a confidencialidade de informações, como Bell-Lapadula, e integridade de informações, como o Biba e o modelo Clark-Wilson.

O conceito de controle de acesso obrigatório foi formalizado primeiramente no modelo Bell-Lapadula [SANDHU 1993]. O modelo BLP une as características de um controle de acesso discricionário com um controle de acesso obrigatório, para a implementação das políticas de segurança que controlam o fluxo de informações em um sistema. Essas políticas são chamadas políticas de multinível. O modelo Biba [BIBA 1977] utiliza os mesmos princípios do modelo Bell Lapadula para definir políticas de multinível que asseguram a integridade de informações.

O modelo Clark Wilson [CLARK e WILSON 1987] aponta requisitos de segurança para aplicações comerciais, preocupando-se com a integridade das informações de um sistema como modificação não autorizada, fraudes e erros.

A utilização dos conceitos de políticas baseadas em papéis é a principal característica dos modelos de segurança baseados em papéis (funções), também chamados de modelos RBAC (*Role-Based Access Control*) [WESTPHALL 2000]. Os modelos podem suportar princípios se segurança como o privilegio mínimo (*least privilege*), separação de tarefas (*separation of duties*) e abstração de dados (*data abstraction*). O privilégio mínimo determina que a uma determinada função são atribuídos apenas os direitos mínimos necessários para a realização das tarefas referentes ao papel. A separação de tarefas determina que haja a existência de papéis de forma mutuamente exclusiva na execução de determinadas atividades, como a existência de um funcionário e um gerente contábil que realizam a emissão de cheques. A abstração de dados permite o uso de direitos como crédito e débito em uma conta, ao invés do uso de direitos como leitura, escrita e execução tipicamente utilizado em sistemas operacionais [SANDHU 1998].

2.5. MECANISMOS DE SEGURANÇA

Os mecanismos de segurança são necessários para implementar políticas expressas por modelos de segurança. Mecanismos de segurança podem ser utilizados em conjunto para atingir um maior grau de satisfação das políticas expressas para um sistema.

Essa satisfação pode ser atingida através do uso do núcleo de segurança em adição a outros controles de segurança. O Núcleo de segurança é implementação do conceito de um monitor de referência. A união do núcleo de segurança a controles adicionais de

segurança, como criptografia, autenticação e identificação, garantem que uma política de segurança possa ser implementada. A união entre esses conceitos é chamada de TCB (*Trusted Computing Base*) definida pelo livro laranja [DOD 1985], do departamento de defesa dos Estados Unidos (DoD).

Mecanismos de segurança são formados pelas diversas técnicas de segurança existentes. Entre os principais mecanismos de segurança pode-se destacar a criptografia, autenticação e controle de acesso.

A criptografia é definida como a ciência da escrita secreta [GOLLMANN 1999]. Pode ser definida também como um conjunto de técnicas que tornam possível que uma informação seja protegida através da transformação da informação original (texto plano) em outra sem sentido lógico (texto cifrado), através de métodos e códigos secretos. Somente quem conhece o método ou o código secreto pode entender o que está contido na informação sem sentido.O processo inverso de cifragem é a decifragem. O uso da criptografia garante a concretização dos aspectos de segurança, como a integridade e confidencialidade.

Algoritmos criptográficos são os mecanismos responsáveis por cifrar e decifrar uma determinada informação. Os algoritmos criptográficos são classificados em algoritmos simétricos e algoritmos assimétricos.

Os algoritmos simétricos caracterizam-se por utilizarem permutações e substituições e uma chave secreta para cifrar uma informação. A chave secreta utilizada nesses algoritmos é a mesma para cifrar e decifrar uma informação. Os principais algoritmos simétricos são [STALLINGS 1999]: o DES (*Data Encryption Standard*) [SMID e BRANSTAD 1988]; o BLOWFISH [SCHN 1993] o RC5 [RIVEST 1995]; e o IDEA (*International Data Encryption Algorithm*) [LAI 1992].

Os algoritmos assimétricos ou de chave pública caracterizam-se por utilizarem um par de chaves, uma chave pública e outra privada, para cifrar uma informação. Um usuário possui um par de chaves: uma privada, que deve se mantida em segredo, e outra chamada de pública que pode ser distribuída para outros usuários. Essas chaves são geradas a partir de cálculos matemáticos, gerando uma chave privada e a partir dela uma chave pública associada, permitindo que a informação cifrada por uma das chaves possa ser decifrada pela outra. A idéia é que um usuário “A” utilize uma chave pública de outro usuário “B” para cifrar uma informação. Assim somente “B” poderá decifrar a

mensagem enviada por “A”, pois só “B” conhece a sua chave privada. A força desses algoritmos depende da dificuldade computacional de se descobrir uma chave privada através da chave pública associada. Quanto maior o tamanho de uma chave maior é a complexidade para calcular seu valor. O principal algoritmo de chave pública é o algoritmo RSA [RIVEST 1978]. Atualmente existem esforços para padronização de algoritmos baseados em criptografia com curvas elípticas (*Elliptic Curve Cryptography ECC*) [STALLINGS 1999].

A autenticação é o processo onde se identifica se uma informação é autêntica. Pode ser usada, por exemplo, para verificar a identidade de um usuário em um sistema computacional ou ainda se uma mensagem enviada por um usuário é legítima e íntegra. Geralmente a autenticação utiliza um método criptográfico para determinar a legitimidade e integridade de informações. Os métodos de autenticação são baseados nos princípios do conhecimento e posse.

A autenticação por conhecimento tem por base a necessidade de um usuário conhecer alguma coisa, por exemplo, uma senha. A autenticação por posse tem por base identificar um usuário através de uma coisa que ele possua, por exemplo, um cartão de identificação. Esse tipo de autenticação pode ser feito através do uso de mecanismos de distribuição de chaves onde o usuário é autenticado em um sistema computacional através de uma chave criptográfica. Usuários podem ser autenticados através do uso de mecanismos de autenticação com chave secreta e mecanismos de autenticação de chave pública. A autenticação por chave pública geralmente é realizada através do uso de certificados digitais, como X.509, onde um usuário é identificado através do seu certificado.

Autenticar um usuário é necessário para controlar o acesso aos recursos os quais ele terá direito de utilizar. O controle de acesso também é um mecanismo de segurança, descrito na seqüência.

2.6. CONTROLE DE ACESSO

Nos conceitos de políticas de segurança se define a funcionalidade de um monitor de referência em um sistema, permitindo ou negando aos sujeitos o acesso requisitado, aos objetos, através da tomada de uma decisão. Esta decisão geralmente é baseada no conjunto de atributos de segurança que definem como um acesso deve ser liberado ou negado. Um projeto de sistema seguro pode adotar qualquer mecanismo que julgue capaz de executar o processo de decisão com base nas políticas de segurança do sistema.

O mecanismo que executa o processo de decisão é chamado de mecanismo de controle de acesso. Em [AMOROSO 1994] controle de acesso é definido como união de mecanismos que executam a mediação em requisições de acesso de sujeitos sobre objetos especificados nas políticas de segurança. A mediação é referenciada como o processo de decisão.

A exemplo das políticas de segurança os mecanismos de controle de acesso são classificados em discricionários, obrigatórios e baseados em papéis [WESTPHALL 2000].

O controle de acesso discricionário (*Discretionary Access Control – DAC*) permite que os usuários possam definir como suas informações podem ser acessadas por outros usuários do sistema. Esse tipo de controle de acesso possui a vantagem da flexibilidade onde o usuário A dono de uma informação determina quem deve acessá-la. A desvantagem é que um usuário de posse de uma informação de A, pode livremente disponibilizar uma copia para outros usuários sem que A perceba.

O controle de acesso obrigatório (*Mandatory Access Control – MAC*) define o acesso às informações de maneira centralizada através de regras incontornáveis. O MAC determina o acesso as informações de acordo com os parâmetros estabelecidos por sistema administrador o que o torna menos flexível que o DAC. Esses parâmetros são determinados através da classificação dos usuários, ou grupos de usuários, em níveis de segurança e rótulos de segurança, o que pode ser também chamado de controle de acesso baseado em reticulados (*lattice-based access control*).

O controle de acesso baseado em papéis (*Role-based Access Control - RBAC*) define o acesso às informações baseadas na classificação dos usuários de acordo com

sua função. Um usuário classificado como secretaria nunca poderá executar ações ou ter acesso a informações a que um gerente tem direito.

2.6.1. Modos e Direitos de Acesso

O modo de acesso é uma definição em um nível mais abstrato das operações de um sujeito sobre um objeto. Os modos de acesso podem ser determinados de maneira diferente de acordo com a necessidade de um sistema. Não existe uma definição específica que determina como deve ser um modo de acesso. Geralmente cada modelo de segurança adota o seu próprio modo de acesso. O modo de acesso pode ser classificado em alteração e observação. A alteração permite que se altere o conteúdo de um objeto. A observação permite que se observe o conteúdo de um objeto.

Assim a maioria das políticas de controle de acesso podem ser expressas em termos de alteração e observação [GOLLMANN 1999]. Geralmente os modos de acesso determinam os direitos de acesso para cada sujeito. Podem existir direitos de acesso como escrever, ler, executar e anexar. Cada um desses direitos pode ser definido a partir da combinação dos modos de segurança. Por exemplo, o direito de leitura pode ser concedido a um sujeito quando ele pode observar o conteúdo de um objeto. Para entender a diferença entre modo de acesso e direito de acesso a figura 1 demonstra como o modelo de segurança Bell-Lapadula determina os direitos de acessos.

Direitos de Acesso **Modos de Acesso**	**Ler**	**Escrever**	**Anexar**	**Executar**
Observação	X	X		
Alteração		X	X	

Figura 1 – Direitos de Acesso no Modelo Bell-Lapadula [GOLLMMAN 1999].

O direito de escrita compreende a alteração e observação de um objeto. A operação de anexar pode ser definida como uma escrita as cegas, já que o sujeito não pode observar o resultado da operação.

2.7. MECANISMOS DE CONTROLE DE ACESSO

2.7.1. Matriz de Controle de Acesso

A matriz de controle de acesso é um modelo conceitual que ajuda a descrever direitos de acessos de sujeitos sobre objetos em um sistema. Cada linha dessa matriz de acesso corresponde a um sujeito e cada coluna corresponde a um objeto. Assim a intersecção entre uma coluna e uma linha determina qual o direito de um sujeito sobre um objeto. A figura 2 demonstra um exemplo de matriz de acesso, onde Ana pode ler e escrever um arquivo que pertence a João enquanto Maria não pode fazê-lo.

Objetos **Sujeitos**	**Arquivo 1**	**Arquivo 2**	**Arquivo 3**	**Programa X**
Ana	Ler, escrever	Ler	Dono	Executar
Maria	-	Dono	Ler	-
João	Dono	-	Ler, escrever	-
Programa X	Ler	Ler	-	-

Figura 2 – Exemplo de Matriz de Acesso.

Da mesma maneira, João não tem nenhum direito sobre um arquivo que pertence a Maria. Mas nada impede que a Ana possa disponibilizar para Maria uma cópia do Arquivo de João. As matrizes de acesso podem ser implementadas através de listas de controle de acesso e *Capabilities*.

2.7.2. Listas de controle de Acesso – ACL

Uma lista de controle de acesso (*Access Control List - ACL*) armazena os direitos de acesso a objetos separadamente para cada objeto. A ACL corresponde a uma coluna da matriz de acesso e estabelece quem pode acessar a dado objeto [GOLLMANN 1999]. A figura 3 mostra exemplos de Listas de Controle de Acesso.

Arquivo 1 : Ana(ler , escrever), João(dono)
Arquivo 2: Ana (ler), Maria (dono) , Programa X (ler)
Arquivo 3 : Ana(dono), Maria (ler), João(ler, escrever)
Programa X: Ana (executar)

Figura 3 – Exemplo de Listas de Controle de Acesso.

A ACL permite que todos os direitos de acesso sobre um objeto sejam facilmente visualizados e revogados. O ato de revogar ou visualizar os direitos de acesso de um único sujeito torna-se mais trabalhoso devido ao fato de que para se conhecer os direitos de um sujeito é necessário efetuar uma busca através de todas as listas de controle de acesso.

2.7.3. Capabilities

Ao contrário de listas de controle de acesso, essa abordagem define os direitos de acesso de sujeitos sobre objetos separadamente para cada sujeito. Cada sujeito possui uma *capability*, que é representada através de uma lista que contém todos objetos e os direitos de acesso correspondentes ao sujeito. A figura 4 mostra um exemplo de listas de *Capabilities*.

Ana: Arquivo1 (ler, escrever), Arquivo2 (ler), Arquivo(dono), ProgramaX(executar)
Maria: Arquivo2 (ler), Arquivo3(dono)
João: Arquivo1(dono), Arquivo3(ler, escrever)
Programa X: Aqruivo1 (ler), Arquivo2 (ler)

Figura 4 – Exemplo de *Capabilities*.

Tipicamente *capabilities* são associadas ao controle de acesso discricionário. Quando um sujeito cria um novo objeto, pode ser dado a outros sujeitos o acesso a esse objeto transferindo a eles a *capability* apropriada [GOLLMANN 1999]. A lista de c*apabilities* permite a visualização e revogação de todos os direitos de um sujeito. Assim determinar quais sujeitos podem acessar um único objeto resulta na inspeção de todas as listas de *capabilities*.

É possível combinar ACL e c*apabilities*. A posse de uma *capability* é suficiente para que um sujeito obtenha acesso autorizado. Por exemplo, em um sistema distribuído a obtenção de uma *capability* permite que um sujeito seja autenticado uma única vez pelo sistema. Excluindo um processo de re-autenticação [SANDHU e SAMARATI 1994].

2.7.4. Rótulos de segurança

Um rótulo de segurança é definido como um atributo associado a entidades de um sistema para denotar seu grau de importância ou sensibilidade. Especificamente, um

rótulo de segurança consiste em dois componentes que são chamados de nível de segurança e categoria de segurança.

O nível de segurança é definido como um atributo hierárquico que pode ser associado a entidades de um sistema computacional para ajudar a definir seu grau de sensibilidade. Por exemplo, em um sistema computacional, um arquivo pode ter um alto grau de importância e outro pode ter um baixo grau de importância [AMOROSO 1994].

Os níveis de segurança podem ter uma conotação própria, como militar ou ainda comercial. De acordo com [AMOROSO 1994] pode-se classificar os níveis de segurança militares em NÃO-CLASSIFICADO, CONFIDENCIAL, SECRETO e ULTRA-SECRETO. E os níveis de segurança em um ambiente comercial em PÚBLICO, SENSSÍVEL, PROPRIETÁRIO e RESTRITO. As categorias de segurança são definidas como um agrupamento de entidades de um sistema de maneira não hierárquica que ajudam a definir o seu grau de sensibilidade. Assim categorias de segurança são grupos de entidades e cujas informações podem ser atribuídos níveis de segurança.

Um objeto classificado com categoria A e nível de segurança SECRETO não poderá ser acessado por um sujeito com categoria B com o mesmo nível de segurança. Essa entidade poderá somente ser acessada por sujeitos com a mesma ou maior habilitação, ou seja, possuir uma mesma categoria e um nível de segurança maior ou igual. O rótulo de um objeto é sua classificação. O rótulo de um sujeito é sua habilitação.

O acesso de uma entidade à outra, que possua um nível de segurança menor, executa-se por meio da chamada relação de dominância.

2.8. CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou os fundamentos básicos da segurança computacional, abordando aspectos relevantes para a compreensão e conhecimentos das técnicas e metodologias aplicadas para a implementação de segurança em ambientes computacionais.

3. WEB SERVICES E XML

Com o crescimento da rede mundial de computadores, cresceu também a necessidade de se estabelecer vários serviços que pudessem atender a esse novo mercado. A necessidade de serviços voltados para usuários da internet causou o surgimento dos *Web Services*. As tecnologias como *Web Services* necessitam de padrões de segurança que possam ser aplicados em seu ambiente, para que usuários possam ter suas informações trafegando entre diversos pontos da rede de maneira segura. Os *Web Services* podem permitir o acesso direto a aplicações, o que pode expor também redes corporativas a várias ameaças de segurança.

3.1. WEB SERVICES

Web Services definem um grupo de padrões que permite que aplicações computacionais se comuniquem e troquem dados através da internet. Embora o verdadeiro impacto dos *Web Services* ainda não seja conhecido, muitos fatores, como fabricantes de softwares que difundem suporte para padrões emergentes, indicam que os *Web Services* estão mudando radicalmente as arquiteturas de TI para efetuar relacionamentos entre parceiros comerciais. Companhias implementam *Web Services* para facilitar uma grande variedade de processos de negócios, como integração de aplicações e transações de B2B.

Aplicações de múltiplas camadas dividem tarefas através de diferentes computadores, por exemplo, uma aplicação de três-camadas pode ter a interface de usuário em um computador, processamento lógico em um segundo e o banco de dados em um terceiro. Para um sistema distribuído funcionar corretamente, os componentes de uma aplicação que estão sendo executados em diferentes computadores através da rede precisam ser capazes de se comunicar. Nos anos 90 muitas organizações desenvolveram tecnologias proprietárias para estabelecer comunicação através de componentes distribuídos. As principais tecnologias são DCOM [EDDON 1998], CORBA [OMG 1999], RMI/IIOP [JURIC 2000]. Essas tecnologias permitem escrever programas em

várias linguagens diferentes, variando as implementações para permitir a execução em diferentes locais. O desenvolvimento dessas tecnologias foi de grande importância para transações e negócios através da *web* por permitirem a integração de entre aplicações.

Porém a interoperabilidade entre essas tecnologias é limitada. *Web Services* podem melhorar a capacidade de computação distribuída com relação a interoperabilidade de tecnologias proprietárias. Assim *Web Services* podem operar usando padrões abertos (*Open-Standards*). Isso significa que *Web Services* podem, teoricamente, permitir a comunicação entre dois componentes de software quaisquer. Organizações podem implementar *Web Services* para melhorar a comunicação entre componentes de tecnologias de comunicação como DCOM e CORBA para criar sistemas distribuídos baseados em padrões.

Com o surgimento o dos *Web Services*, muitos desenvolvedores perceberam que a tecnologia poderia revolucionar a computação distribuída [DEITEL 2002]. Previamente, havia uma expectativa de que CORBA ou DCOM fosse escolhida como padrão universal de computação distribuída.

A experiência industrial com os problemas de interoperabilidade conduziram ao desenvolvimento de padrões abertos para tecnologia de *Web Services*, em um esforço para tornar viável a comunicação entre diferentes plataformas. O primeiro padrão desenvolvido para esta comunicação foi o XML (*eXtensible Markup Language*), uma linguagem que utiliza marcação de texto, que permite a troca de informações entre aplicações e plataformas.

Um protocolo de transporte de informações e instruções entre *Web Services*, desenvolvido pela Microsoft e DevelopMentor, é o protocolo SOAP [BOX 2000] (*Simple Object Access Protocol*), baseado no padrão XML. Outras especificações de *Web Services* que também são baseados em XML são a WSDL [CHRISTENSEN 2001] (*Web Services Description Language*) e a UDDI [BELLWOOD 2002] (*Universal Description Discovery and Integration*). A WSDL provê um método padrão para descrever *Web Services* e suas propriedades (*Capabilities*). A UDDI define regras para construção de diretórios nos quais empresas anunciam seus *Web Services*.

Assim os *Web Services* tem como vantagens utilizar padrões abertos, permitindo que componentes sejam escritos para diferentes linguagens e plataformas. *Web Services* podem ser facilmente implementados e possuem baixo custo de implementação, pelo

fato de utilizar estruturas e protocolos de comunicação existentes para efetuar troca de informação. *Web Services* podem reduzir os custos de integração entre aplicações como B2B.

Além de oferecer vários benefícios, os *Web Services* também criam vários desafios para desenvolvedores. Entre eles o fato de que *Web Services* não possuem ainda padrões de procedimentos de segurança. *Web Services* tipicamente permitem o acesso direto a aplicações, o que pode expor redes corporativas a ameaças de segurança como *hackers* e vírus. A maioria dos padrões desenvolvidos para *Web Services*, como SOAP, não foram projetados para fornecer segurança. Assim, algumas especificações de segurança estão sendo desenvolvidas e padronizadas por organizações como a W3C (*World Wide Web Consortium*) e OASIS. Os padrões de segurança para *Web Services* utilizam também como base o padrão XML. Os principais padrões de segurança para *Web Services* com base em XML são *XML-Signature*, para integridade e assinaturas; o *XML Encryption*, para confidencialidade; o *XML Key Management* (XKMS), para gerenciamento de chaves; a *Security Assertion Markup Language* (SAML), para autenticação e autorização; a *eXtensible Access Control Markup Language* (XACML), para estabelecer como definir as políticas de segurança para o controle de acesso.

A implantação e implementação dessas novas tecnologias podem oferecer novas perspectivas para organizações com relação a segurança entre negócios e transações comerciais através da *web*.

3.2. XML (*EXTENSIBLE MARKUP LANGUAGE*)

O XML (*eXtensible Markup Language*) [BRAY 2000] é uma linguagem de marcação desenvolvida pela W3C (*World Wide Web Consortium*), inicialmente para sobrepor algumas limitações do HTML (*Hyper Text Markup Language*), e oferecer suporte a interoperabilidade de uma grande variedade de aplicações *web*. W3C é a organização encarregada do desenvolvimento e manutenção da maioria dos padrões *web*. O XML e o HTML são padrões baseados em SGML (*Structured General Markup Language*), onde a estrutura e formato de documentos são determinados através do uso de elementos marcadores.

O HTML é a mais popular linguagem de marcação e é suportada por milhares de aplicações de diferentes níveis e segmentos em sistemas computacionais, como por exemplo, *browsers*, aplicações de bancos de dados e softwares de *email*. O HTML foi

aprimorada com o passar do tempo buscando sua melhoria e adaptação para suprir as necessidades de aplicações e *Web Services*. Porém, aplicações *web* podem tornar-se muito flexíveis e algumas podem exigir que a tecnologia usada, na disposição das informações, tenha que sofrer algumas alterações de acordo com suas necessidades. O padrão HTML possui um conjunto de marcadores, também chamados de *tags*, fixos e que determinam a estrutura e apresentação de um documento sem muita flexibilidade. Já o padrão XML em sua principal característica não descreve a apresentação de um documento e sim o seu conteúdo. O XML é extensível, ou seja, permite a definição de marcadores de acordo com a necessidade imposta. O XML ainda permite que seus marcadores sejam definidos em infinitos níveis de profundidade. Assim o padrão XML vem substituir o HTML na disposição de conteúdo em virtude da sua flexibilidade e extensibilidade necessárias para aplicações e *Web Services*.

Os principais objetivos traçados para o padrão XML estabelecidos pela W3C determinam que o padrão XML deve ser diretamente utilizável e aproveitável através da internet; o XML deve suportar uma grande variedade de aplicações; o XML deve ser compatível com SGML; o XML deve ser fácil de ser escrito e interpretado por programas que processam XML; documentos XML precisam ser legíveis e razoavelmente claros; documentos XML devem ser fáceis de criar; o projeto XML deve ser formal e conciso.

Com isso pode-se destacar algumas aplicações e áreas onde o XML pode ser utilizado, como em manutenção de grandes *Web Sites*, troca de Informações entre organizações, aplicações de comércio eletrônico onde organizações colaboram para servir aos seus consumidores, distribuição de livros e documentos eletrônicos com linguagens de marcação para expressar os direitos sobre o conteúdo e também como controle para distribuição de informações de maneira segura. Segundo [SIMEON 2002] existem vários sistemas de definição de tipos para XML, incluindo DTDs, recomendação original do W3C, XML *Schema* [BEECH 2001] como recomendação para substituição do DTD pela W3C e Relax NG [MURATA 2001], padrão desenvolvido pela Oasis. A W3C é também responsável por três linguagens de programação ligadas como XML: a XSLT uma linguagem para apresentação e transformação de documentos; a *Xquery*, uma analogia ao SQL para dados XML; e o *XPath*. O *XPath* é uma linguagem para endereçamento de partes de um documento

XML. O uso do *XPath* facilita o acesso direto aos elementos de um documento XML, não necessitando a navegação no documento. Todas essas são linguagens funcionais. XSLT 1.0 [CLARK 1999b] e o *XPath* 1.0 [CLARK 1999a] tornaram-se recomendações da W3C em novembro de 1999. XML *Schema* tornou-se uma recomendação em maio de 2001. XSLT 2.0, *XQuery* 1.0 e o *XPath* 2.0 estão sendo desenvolvidos.

3.2.1. Estrutura lógica de Documentos XML

Um documento XML é formado por uma seqüência de elementos aninhados, cada um delimitado por um par de marcadores denominados *tags*, um para o início e outro para o fim (e.g., <Exame> e </Exame>) ou por uma *tag* vazia. Cada elemento pode dar origem a uma cadeia de vários outros elementos, formando uma cadeia de elementos que pode ser chamada de árvore de elementos do documento XML. Assim cada nó da árvore é formado por um elemento podendo conter outros nós filhos. Isso possibilita que a estrutura de um documento possa atingir qualquer grau de profundidade, de acordo com a complexidade que documentos possam exigir em sua estrutura.

```
<?xml version="1.0" encoding="UTF-8"?>
<Exame>
   <Resultado ID="Rosset">
      <Sangue>
            <PLaquetas>0.5</PLaquetas>
            <Hemoglobina>0.12</Hemoglobina>
      </Sangue>
   </Resultado>
</Exame>
```

Figura 5 – Exemplo de documento XML.

A figura 5 apresenta um exemplo de um documento XML que representa um resultado de um exame de sangue de um paciente. È importante ressaltar que todo documento XML possui em seu cabeçalho as informações referentes a versão que o documento XML foi escrito. Documentos XML possuem um elemento raiz que dá início à árvore de elementos em sua estrutura, representado na figura 5 pelo elemento <Exame>.

O padrão XML permite a definição de atributos para elementos. Os atributos representam as propriedades dos elementos. Um elemento pode possuir vários atributos, e cada atributo possui um único valor. Os valores de atributos são delimitados por aspas duplas (“). Como exemplo, na figura 1, o elemento <Resultado> possui um atributo ID que possui o valor (Rosset). Sub-elementos também podem ser vistos como atributos de

um elemento. Por exemplo, os sub-elementos <Plaquetas> e <Hemoglobina> podem ser interpretados como atributos pertencentes ao elemento <Sangue>. Os elementos permitem definir estruturas mais complexas que os atributos. Os elementos podem conter sub-elementos e podem ser utilizados para definir objetos e estruturas complexas.

3.2.2. Definição de Gramáticas e Validação de Documentos

Documentos XML podem ser classificados por sua formatação e gramática, podendo ser bem-formados (*well-formed*) e válidos (*valid*) respectivamente. Um documento XML é considerado bem formado quando ele obedece à sintaxe do XML, não possuindo *tags* vazias ou quando um elemento com marcação de início não possui marcação de final.

Um documento bem formado é considerado válido quando ele obedece à estrutura determinada por uma gramática que pode ser expressa através de uma *Document Type Definition* (DTD). Uma DTD contém a definição dos tipos de elementos e atributos que fazem parte do documento XML. Uma DTD pode incluir declarações para elementos, atributos, entidades e comentários. Os elementos [DAMIANI 2000] são os componentes mais importantes de um documento XML.

A declaração de elementos em uma DTD (figura 6) especifica o nome dos elementos e seu conteúdo. Ela também pode descrever sub-elementos na sua estrutura utilizando uma sintaxe formada de símbolos chamados de indicadores de ocorrência e conectores. O símbolo “*” indica nenhuma ou mais ocorrências, já o “+” indica uma ou mais ocorrências, o “?” indica nenhuma ou uma ocorrência e quando não houver nenhum indicador assume-se apenas uma ocorrência daquele elemento em todo o documento.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Exame (Resultado*)>
<!ELEMENT Resultado (Sangue?)>
<!ATTLIST Resultado ID ID #REQUIRED>
<!ELEMENT Sangue (PLaquetas?, Hemoglobina?)>
<!ELEMENT Hemoglobina (#PCDATA)>
<!ELEMENT PLaquetas (#PCDATA)>
```

Figura 6 – Exemplo de DTD.

Documentos construídos através da DTD possuem algumas limitações. Uma DTD impõe ordem aos elementos de um documento XML, e não especifica tipos de maneira abrangente. Uma DTD permite a construção de documentos complexos, mas o

acervo de tipos se limita ao PCDATA, ou seja, seqüências de caracteres. Assim não é possível a definição de tipos de dados como inteiros, datas, horas, nem números de ponto flutuante. Através do DTD não é possível definir um intervalo de valores para um tipo, nem limitar o número de ocorrências de um sub-elemento dentro de outro.

A DTD não é a única maneira para definir um documento em XML. Para isso, também podem ser utilizados outros padrões para especificação de gramáticas como XML *Schema* [BEECH 2001], o padrão XML-Data [LAYMAN 1998], proposto pela ArborText, DataChannel, Inso e Microsoft, o padrão RDF [BRICKLEY 2000] (*Resource Description Format*), o padrão DCD [FRANKSTON 1998] (*Document Content Description*) proposto pela IBM, Microsoft e a Textuality, que permitem a especificação de elementos globais e locais bem como todas as características da DTD. Os quatro padrões descritos são mais fáceis de serem compreendidos por uma leitura visual. Eles são expressos em XML, e não em uma linguagem diferente, como é o DTD.

Tal como os padrões anteriores, o XML *Schema* apresenta novas características que o diferenciam do modelo de esquema da DTD como tipos de dados (e uma variedade de tipos de dados pré-definidos), declaração de intervalo de valores para tipos de dados e cardinalidade de atributos dentro de um elemento.

Um conceito interessante do XML *Schema* é o de grupos de atributos, que permite o compartilhamento de grupos de atributos entre diferentes elementos. O XML *Schema* torna-se mais atraente que o DTD quando os documentos XML tornam-se maiores e mais complexos. Da mesma maneira podem existir casos em que os documentos são menores e possuem uma estrutura mais simples, neste caso pode-se utilizar DTD. A figura 7 apresenta um exemplo de XML *Schema*.

Sendo assim, a escrita de um documento XML deve seguir as determinações estabelecidas por uma gramática através de um padrão como DTD ou *Schema* referente a cada documento, para que ele possa ser considerado válido e conseqüentemente bem formado.

```
<complexType name="ResponseAbstractType" abstract="true"> <sequence>
<element ref = "ds:Signature" minOccurs="0"/>
</sequence>
<attribute name="ResponseID" type="ID" use="required"/>
<attribute name="InResponseTo" type="NCName" use="optional"/>
<attribute name="MajorVersion" type="integer" use="required"/>
<attribute name="MinorVersion" type="integer" use="required"/>
<attribute name="IssueInstant" type="dateTime" use="required"/>
<attribute name="Recipient" type="anyURI" use="optional"/>
</complexType>
```

Figura 7 – Exemplo de XML Schema.

3.2.3. Espaços de nomes (Namespaces)

Namespaces ou espaços de nome podem ser definidos como mecanismos de identificação de elementos escritos em XML. A definição de *namespaces* [BRAY 1999] surgiu da necessidade de diferenciar *tags* com mesmo nome em documentos XML.

A livre criação e ordenação de *tags* em diversas camadas da árvore de elementos de documentos XML é o que da a essa linguagem sua principal característica que é a flexibilidade. A utilização de *tags* que possuem o mesmo nome em um mesmo nível da árvore do XML pode provocar uma confusão, por exemplo, pelo fato de uma aplicação não poder determinar em qual nó está a informação que se deseja, por que podem existir dois ou mais nós com o mesmo nome. Por outro lado restringir a duplicação de nomes acarretaria em restringir a flexibilidade do XML.

Assim XML *namespaces* são utilizados para diferenciar e identificar elementos com mesmo nome que pertencem a domínios diferentes. Para isso são utilizados prefixos que identificam a qual domínio um elemento pertence. A figura 8 apresenta um exemplo de duplicação de nós onde elementos com o mesmo nome precisam ser diferenciados para se obter o resultado correto de um exame.

```
<?xml version="1.0" encoding="UTF-8"?>
<Exame>
 <Resultado>
        <Sangue></Sangue>
 </Resultado>
 <Resultado>
        <Gravidez></Gravidez>
 </Resultado>
</Exame>
```

Figura 8 – Exemplo de XML com dados duplicados.

A figura 9 mostra a estrutura do mesmo documento utilizando XML *namespaces* para diferenciar os elementos no mesmo documento.

```
<?xml version="1.0" encoding="UTF-8"?>
<Exame>
<references xmlns:pe=”http://www.hospital.com.br/pediatria”
xmlns:cg=” http://www.hospital.com.br/clinica”>
 <cg:Resultado>
            <Sangue></Sangue>
 </cg:Resultado>
 <pe:Resultado>
            <Gravidez></Gravidez>
 </pe:Resultado>

</Exame>
```

Figura 9 – Exemplo de *namespace* em XML.

A utilização dos prefixos *pe* e *cg* diferenciam os elementos como mesmo nome do documento XML. Não são os prefixos que determinam a unicidade de um nome no XML *namespaces* e sim a URI a qual um prefixo aponta. XML *namespaces* utiliza URI para identificar elementos, porque elas são baseadas nomes de domínio, que são registrados e únicos.

3.2.4. Manipulação de documentos XML

A criação de documentos XML deve seguir as especificações contidas em uma gramática estabelecida e a sintaxe do padrão XML. Quando se escreve ou se lê um arquivo que contenha um documento XML, deve-se verificar a consistência do mesmo em relação a gramática e a sintaxe, ou seja, deve-se verificar se o documento é valido e bem formado. Essa verificação pode ser feita com a utilização de uma ferramenta de auxilio chamada de *parser* XML. Um *parser* XML pode funcionar como um validador de documentos XML, verificando a estrutura do documento de acordo com as especificações de uma DTD ou *Schema*. Assim a característica de validação de documentos é o que divide os *parsers* em dois grupos os *parsers* de validação e os *parsers* de não validação.

Aplicações armazenam as modificações em documentos XML através dos *parsers*. Existem dois modos de conectar a aplicação com o *parser*: através do uso de interfaces baseadas em objetos e interfaces baseadas em eventos. Quando um *parser* utiliza uma interface baseada em objetos, ele resgata toda árvore de nós de um documento XML, de forma que cada elemento corresponde a um objeto, formando uma árvore de objetos e a disponibiliza na memória. A aplicação não trabalha direto com o

documento XML e sim com um espelho que o *parser* disponibiliza. Quando um *parser* utiliza uma interface baseada em eventos ele lê o documento e gera eventos que determinam elementos, atributos e texto do documento XML, interfaces baseadas em eventos trabalham direto com o documento. Os dois métodos não são concorrentes e sim complementares e utilizados para fins diferentes. Interfaces baseadas em objetos são ideais para aplicações que manipulam documentos XML, e não conhecem a estrutura desses documentos, como editores e *browsers*. Interfaces baseadas em eventos são utilizadas por aplicações que utilizam sua própria estrutura de dados, não sendo necessário carregar uma árvore de elementos na memória.

Os padrões estabelecidos para essas duas abordagens são SAX [MEGGINSON 2000] e o DOM [APPARAO 1998]. O padrão para interface baseada em eventos é o SAX (*Simple API for XML*) que foi desenvolvido por um grupo de desenvolvedores em XML da XML-DEV *mailing list* e editado por David Megginson. O padrão para interfaces baseadas em objetos é o DOM (*Document Object Model*) que foi desenvolvido e padronizado pelo W3C. Tanto o DOM como SAX são disponibilizadas em APIs que facilitam a manipulação de documentos XML.

Geralmente aplicações de manipulação de documentos XML utilizam em conjunto o *parser* e as APIs de manipulação, além de também poder fazer uso de outros padrões para facilitar o seu trabalho. Por exemplo, através de uma aplicação que requisita um documento ou parte de um documento XML, a API de manipulação utilizada aciona o *parser* que se encarrega de validar o documento requisitado, através da DTD ou *Schema* do documento. Validado o documento, a API pode disponibilizar para a aplicação todos os dados requisitados, sejam eles um conjunto de nós ou um único elemento do documento XML. Em uma abordagem baseada em objetos, para facilitar a localização de elementos e acesso a partes de documentos, pode ser utilizado *XPath*. O uso do *XPath* facilita o acesso direto aos elementos de um documento XML, não necessitando a navegação no documento.

Finalmente, de posse dos dados a aplicação pode ainda convertê-los em um formato desejado pelo usuário, através de uma linguagem de transformação de documentos XML, como o XSLT.

3.3. PADRÕES DE SEGURANÇA XML

Web Services podem permitir o acesso direto a aplicações, o que pode expor redes corporativas a várias ameaças de segurança. O uso dessas tecnologias depende do nível de segurança que elas podem oferecer aos usuários, que poderão muitas vezes ter suas informações trafegando entre diversos pontos da *web*.

Os padrões de segurança XML fornecem uma série de especificações para suprir requisitos de segurança. Esses padrões de segurança definem vocabulários em XML para representar informações seguras. Os padrões de segurança permitem que a segurança seja aplicada em documentos XML, em elementos do documento, bem como no seu conteúdo.

Os principais padrões de segurança para *Web Services* com base em XML são o *XML-Signature* [EASTLAKE 2002a]; o *XML Encryption* [EASTLAKE 2002b]; o *XML Key Management* (XKMS) [HALLAM-BAKER 2003]; a *Security Assertion Markup Language* (SAML) [MALER 2002]; a *eXtensible Access Conrol Markup Language* (XACML) [GODIK 2003].

3.3.1. XML-Signature

O padrão XML-*Signature* [EASTLAKE 2002a] foi especificado pelo W3C e é utilizado para representar assinaturas digitais. Essa especificação se aplica a requisitos de segurança para a autenticação de documentos digitais, verificação da integridade de documentos digitais e também a não-repudiação de documentos, uma vez que esses documentos foram assinados. Os algoritmos utilizados nessa especificação incluem o algoritmo de chave pública *Digital Signature Standard* (DSS) e o algoritmo de autenticação *Secure Hash* (SHA-1). Todavia, desenvolvedores podem utilizar o XML-*Signature* para suportar seus próprios algoritmos e modelos de segurança. XML-*Signature* pode ser utilizado para assinar qualquer tipo de arquivo, não somente documentos XML, e os dados assinados podem estar dentro ou fora do documento XML que contém uma assinatura. São utilizadas URIs para associar as assinaturas aos objetos de dados assinados.

XML-*Signature* suporta a utilização de múltiplas assinaturas em um documento XML, bem como para diferentes seções de um documento. XML-*Signature* especifica técnicas relacionadas com a verificação de assinatura com relação a variação de

estrutura de documentos XML. XML-*Signature* assina documentos XML que estão na chamada forma canônica. Antes de um documento ser assinado ele deve ser colocado em sua forma canônica, isso é feito através do uso do XML *Canonicalization* [BOYER 2001], que transforma documentos com o mesmo conteúdo e estruturas diferentes em documentos idênticos. Colocar um documento em sua forma canônica é necessário porque quando se utilizam algoritmos de *Hash*, diferentes entradas podem produzir diferentes resultados. O resultado do cálculo efetuado por algoritmos de *Hash* é chamado de *Digest*. XML *Canonicalization* é utilizado para reduzir variações em documentos para que haja interoperabilidade entre aplicações *web*. Pode ser possível que dois ou mais documentos possuam o mesmo conteúdo diferindo apenas na estrutura do mesmo, por exemplo, um espaço em branco. A figura 10 apresenta um exemplo de um Documento XML-*Signature*.

```
01 <Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
02   <SignedInfo>
03   <CanonicalizationMethod   Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
04   <SignatureMethod   Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
05   <Reference         URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
06     <Transforms>
07       <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
                c14n-20010315"/>
08     </Transforms>
09     <DigestMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
10     <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
11   </Reference>
12 </SignedInfo>
13   <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
14   <KeyInfo>
15a    <KeyValue>
15b      <DSAKeyValue>
15c        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
15d      </DSAKeyValue>
15e    </KeyValue>
16   </KeyInfo>
17 </Signature>
```

Figura 10 – Exemplo de XML-*Signature*.

O elemento <*SignedInfo*> (linha 02 até 12) carrega informações referentes a assinatura do documento. O elemento <*CanonizationMethod*> (linha 03) especifica o algoritmo XML *Canonicalization* utilizado antes de gerar a assinatura do documento. O elemento <*SignatureMethod*> (linha 04) especifica qual o algoritmo utilizado para gerar a assinatura do documento do elemento <*SignedInfo*> em sua forma canônica no valor do elemento <*SignadeValue*> (linha 13) que é a assinatura do documento. O elemento <*Reference*> (linha 5 até 11) especifica informações como o algoritmo para o

cálculo do d*igest* <*DigestMethod*> (linha 9) e o valor do *digest* <*DigestValue*> (linha 10). O elemento <*KeyInfo*> identifica a chave utilizada para validar a assinatura do documento. Essa identificação, por exemplo, pode ser feita através de certificados.

3.3.2. XML Encryption

O padrão XML *Encryption* [EASTLAKE 2002b], definido pelo W3C, apresenta um vocabulário para garantir a confidencialidade do conteúdo de documentos XML através do uso de algoritmos criptográficos. Esse padrão é utilizado com o propósito de manter a confidencialidade de informações que estão em trânsito ou armazenadas. XML *Encryption* permite que diferentes elementos de um documento sejam cifrados separadamente, ou apenas partes de um documento XML sejam cifradas. Um proprietário de um documento (figura 11) pode cifrá-lo para torná-lo confidencial (figura 12), para isso pode-se utilizar, por exemplo, um algoritmo simétrico de criptografia. A figura 11 apresenta uma informação de pagamento que contendo o nome e numero do cartão de credito do usuário “John Smith”.

```
<?xml version='1.0'?>
 <PaymentInfo xmlns='http://example.org/paymentv2'>
   <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Figura 11 – Exemplo de documento XML com dados a serem cifrados [EASTLAKE 2002b]

Quando um elemento XML ou o conteúdo do elemento é cifrado (figura 12), ele é substituído por um elemento denominado <*EncryptedData*>. Na figura 12 o elemento <*CipherValue*> contém o conteúdo cifrado do elemento <*CerditCard*> presente na figura 11. O documento também pode conter um elemento define qual o algoritmo utilizado para cifrar o conteúdo e também alguma informação que indique qual chave é necessária para a decifrar o conteúdo do elemento.

```
<PaymentInfo xmlns='http://example.org/paymentv2'>
   <Name>John Smith</Name>
<EncryptedData
   Type='http://www.w3.org/2001/04/xmlenc#Element'
   xmlns='http://www.w3.org/2001/04/xmlenc#'>
   <CipherData>
      <CipherValue>A23B45C56</CipherValue>
   </CipherData> </EncryptedData></PaymentInfo>
```

Figura 12 – Exemplo de documento com conteúdo cifrado [EASTLAKE 2002b].

3.3.3. XKMS (XML Key Management Specification)

O padrão XML *Key Management Specification* (XKMS) define um protocolo para registrar e distribuir chaves criptográficas em *Web Services* que utilizam infraestrutura de chave pública (PKI). O XKMS foi inicialmente desenvolvido pela Microsoft em conjunto com a VeriSign, depois tornou-se uma iniciativa do W3C. O gerenciamento de chave pública inclui a criação de par de chaves pública e privada, a ligação dessas chaves com uma identidade e disponibilização dessas chaves em diferentes formatos. O XKMS foi criado para auxiliar a distribuição de chaves públicas, que são essenciais para XML *Signature* e XML *Encryption*, possibilitando verificações de assinaturas e criptografia em documentos XML.

O padrão XKMS é dividido em duas especificações a XML *Key Registration Service Specification* (X-KRSS) e o XML *Key Information Service Specification* (X-KISS) [HALLAM-BAKER 2003]. No gerenciamento de chave pública é necessário que ocorra um registro de chaves onde chaves são atribuídas a identidades quaisquer. Também é necessário que seja possível revogar essa associação com as chaves caso ocorra, por exemplo, um roubo da chave privada. Para o registro de chaves é utilizada a XML *Key Registration Service Specification* (X-KRSS), que apresenta um conjunto de protocolos para o gerenciamento de certificados para o registro, revogação, e recuperação de certificados digitais. Um usuário pode registrar um certificado enviando sua chave pública ao servidor de registro confiável através de uma requisição digital assinada com sua chave privada. Essa requisição pode conter informações como nome e atributos.

A especificação X-KISS apresenta um conjunto de protocolos para requisição e distribuição de informações referentes às chaves públicas registradas. X-KISS localiza as chaves públicas e relaciona informações de usuários com elas [DEITEL 2002]. Assim, por exemplo, quando uma aplicação recebe um documento XML assinado com XML *Signature*, ela pode enviar uma requisição X-KISS de verificação de assinatura para um servidor de confiança, que processa a informação. Esse servidor define se a assinatura é válida e libera a aplicação requisitante da tarefa de executar processos lógicos complexos gerados através de algoritmos criptográficos. Depois, o servidor retorna para a aplicação requisitante um documento X-KISS de resposta.

3.3.4. SAML (Secure Assertion Markup Language)

O SAML (*Secure Assertion Markup Language)* [MALER 2002] é um padrão para a transferência de informações de autorização e autenticação através da internet. O SAML é desenvolvido pelo OASIS *Security Services Technical Commitee* (SSTC) como um padrão XML para comunicação em plataformas de B2B (*Business-To-Business*) e B2C (*Business-To-Consumer*). O SAML pode ser definido como uma Infraestrutura de Gerenciamento de Permissões (*Permissions Management Infrastructure - PMI*), e utiliza um conjunto de políticas de segurança para definir um controle de acesso e autorização sobre informações e outros recursos disponíveis em um sistema computacional.

O padrão SAML apresenta um vocabulário para expressar asserções de autorizações e autenticação. Asserções são distribuídas por autoridades SAML, isto é, autoridades de autenticação, autoridades de atributos e PDP (*Policy Decision Point*). O SAML também define um protocolo através do qual clientes podem requisitar asserções à autoridades SAML e receber a resposta para essas asserções.

SAML foi desenvolvido para prover interoperabilidade entre aplicações e pode ser utilizado em conjunto com protocolos de comunicação XML como SOAP [BOX 2000] e ebXML [EBXML 2001]. O padrão SAML permite que usuários enviem as informações para autenticação apenas uma única vez, e sejam autenticados através de vários domínios. Esse processo de autenticação única é chamado de *Single Sign-On* (SSO).

A figura 13 demonstra um modelo conceitual proposto no padrão SAML, onde se pode observar o uso de asserções SAML quando uma entidade do sistema interage com as autoridades de autenticação, autorização e atributos. Neste modelo, a autoridade de autorização pode ser formada pelos componentes [MALER 2002]: *Policy Enforcement Point* (PEP) e o *Policy Decision Point* (PDP). O PEP é a entidade que faz o controle de acesso fazendo requisições de decisão e executando decisões de autorização. O PDP é a entidade que avalia as políticas aplicáveis a sujeitos e define as decisões de autorização.

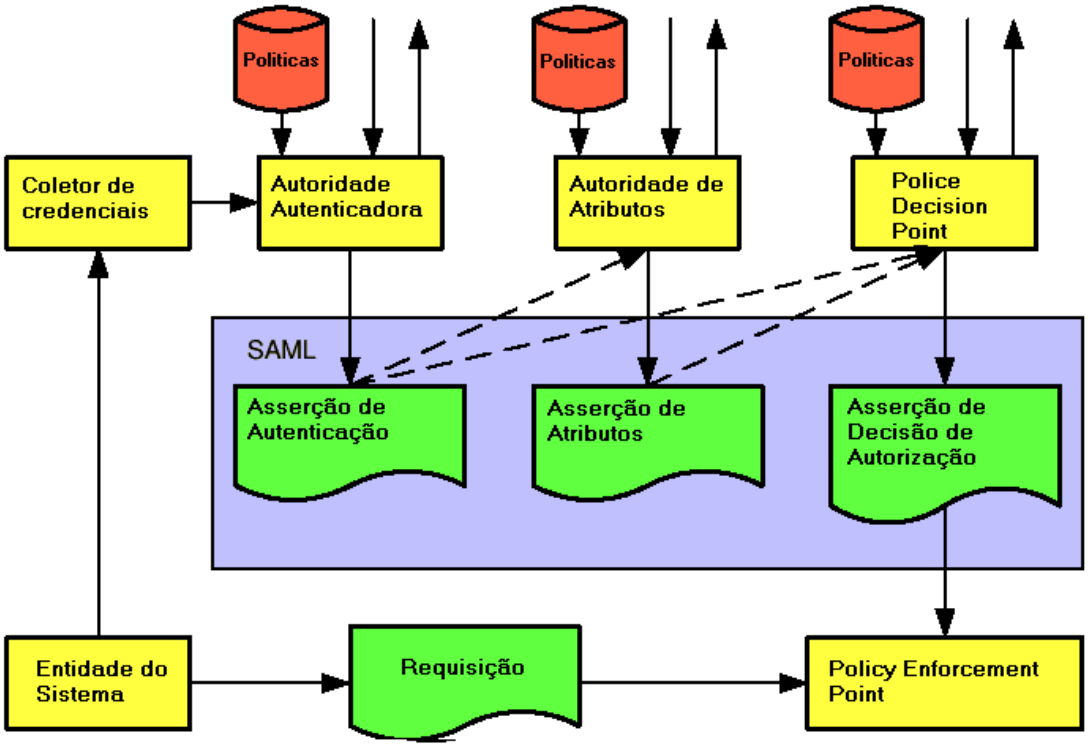

Figura 13 – Modelo de Domínio SAML [MALER 2002]

Asserção é um pacote de informação que carrega uma ou mais declarações de uma autoridade SAML. O padrão SAML define três tipos de asserções que podem ser criados por uma autoridade SAML: na asserção de autenticação o sujeito especificado foi autenticado para um meio particular em um momento particular; na asserção de atributo o sujeito especificado está associado com atributos fornecidos; e na asserção de decisão de autorização é determinada uma decisão de acesso ao sujeito sobre um recurso específico.

Assim asserções podem conter informações sobre ações de autenticação, atributos de sujeitos e decisões de autorização sobre permissões aos sujeitos para acesso a certos recursos. Uma asserção simples pode conter diversas declarações internas sobre autorização, autenticação e atributos [MALER 2003].

Uma declaração de autorização SAML pode possuir campos para identificação da declaração como ID, nome do sujeito, domínio de segurança que o sujeito pertence, informações como a hora da emissão e quem foi o emissor e as condições para as quais a asserção é válida.

Asserções SAML podem ser geradas e enviadas usando uma variedade de protocolos. O padrão SAML define um protocolo para a troca de mensagens chamado

de SAML *request-response*, definido em mensagens SAML pelos elementos <*Request*> e <*Response*>. Um requisitante envia uma mensagem contendo elemento <*Request*> a uma autoridade SAML. A autoridade SAML responde enviando uma mensagem contendo um elemento <*Response*>. As asserções SAML são encapsuladas em *responses.* A figura 14 apresenta um exemplo de asserção de autenticação SAML para um usuário "joão" que pertence ao um domínio "www.company.com".

```
<Assertion AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2003-04-17T00:46:02Z"
        Issuer="www.servidorautenticacao.com"
        MajorVersion="1"
        MinorVersion="1">
 <Conditions
  NotBefore="2003-04-17T00:46:02Z"
  NotOnOrAfter="2003-04-17T00:51:02Z">
 </Conditions>

 <AuthenticationStatement
                        AuthenticationMethod="password"
                        AuthenticationInstant="2003-04-17T00:46:02Z">
            <Subject>
                        <NameIdentifier
                                    SecurityDomain="www.company.com"
                                    Name="joao"/>
            </Subject>
 </ AuthenticationStatement >
</Assertion>
```

Figura 14 – Exemplo de Asserção de Autenticação SAML.

3.3.5. XACML (eXtensible Access Control Markup Language)

A *eXtensible Access Control Markup Language* (XACML) [GODIK 2003] foi desenvolvida pelo OASIS *Security Services Technical Commitee* (SSTC), com fundamentos no XML *Access Control Language* (XACL) da IBM [HADA 2000a] e XML-AC [DAMIANI 2002] da Universidade de Milão. O padrão XACML define linguagens de marcação que permitem especificar políticas de segurança, requisições e respostas para decisões de controle de acesso, permitindo a organizações utilizarem essas políticas para controlar acesso a conteúdos e informações protegidas.

A linguagem para definição de políticas XACML é utilizada para descrever requisitos gerais de controle de acesso, e possui pontos de extensão para definição de novas funções, tipos de dados e combinações lógicas. As políticas de segurança XACML podem controlar o acesso a informação utilizando identidade de clientes, o método de autenticação de clientes ou ainda uma porta pela qual um cliente se comunica.

A linguagem de requisição e resposta (*request-response language*), define como formar requisições de acesso e interpretar as respectivas respostas. Uma resposta sempre determina se a requisição de acesso foi aceita ou não, utilizando um valor definido: *Permit, Deny, Indeterminate* e *Not Applicable*.

O XACML difere de outras linguagens e padrões proprietários primeiramente pelo fato de ser um padrão aberto (*Open-Standard)*. Segundo, por ser genérico, permite que possa ser usado para prover controle de acesso para sistemas completos bem como a um recurso específico. Finalmente por ser aplicável em conjunto com outros padrões como o SAML e LDAP [JOHNER 1998], podendo formar a base para tomada de decisões. O XACML pode ainda ser utilizado em conjunto com *Digital Rights Management* (DRM), utilizando suas políticas para definir privilégios de acesso para usuários. Por exemplo, um distribuidor de livros eletrônicos pode utilizar políticas XACML para permitir que qualquer pessoa possa visualizar um primeiro capítulo de uma obra, e permitir que apenas usuários registrados possam visualizar a obra por completo.

O uso do XACML pode ser feito tanto em ambientes e aplicações proprietárias quanto públicas, podendo facilitar o processo de tomada de decisões e proporcionar interoperabilidade entre diferentes plataformas e domínios.

3.3.6. Arquitetura de autorização XACML

O ambiente especificado no padrão XACML [GODIK 2003] prevê um modelo de arquitetura de autorização baseado na recomendação para um *framework* de autorização da IETF [VOLLBRECHT 2000]. Essa arquitetura utiliza entidades denominadas PEP (*Policy Enforcement Point*), PDP (*Policy Decision Point*), PIP (*Policy Information Point*) e PAP (*Policy Administration Point*). O PEP é a entidade responsável por receber requisições de acesso e aplicar decisões de acesso. O PDP é entidade responsável por tomar decisões de acesso baseadas em políticas de autorização existentes e repassar a decisão para o PEP. O PIP é a entidade que funciona como um repositório de atributos e os disponibiliza para o PDP. O PAP é a entidade responsável por criar as políticas de autorização do sistema e disponibilizá-las para o PDP.

Um exemplo típico do funcionamento desse modelo (figura 15) ocorre quando um usuário já autenticado requisita acesso a um recurso qualquer do sistema (passo 2). Inicialmente o PEP intercepta a requisição de acesso. Depois o PEP forma uma

requisição de autorização em um formato nativo conhecido (passo 3), por exemplo SAML. A entidade manipulador de contexto é responsável por transformar essa requisição de autorização em uma requisição de decisão de autorização XACML e a envia para o PDP (passo 8). O manipulador de contexto pode incluir atributos extras necessários para a formação da requisição de decisão de autorização. Para isso, pode-se requisitar estes valores de atributos para o PIP (passos 4,5,6,7). O PDP ao receber a requisição de tomada de decisão, requisita ao PAP as políticas referentes aos valores de atributos da requisição (passo 1). O PDP de posse das políticas compara os valores contidos em suas condições com os valores requisitados e toma a decisão de acesso. Depois o PDP envia a decisão de autorização XACML para a entidade manipulador de contexto (passo 9) que repassa, em um formato nativo, para o PEP (passo 10). Finalmente o PEP aplica a decisão de autorização ao usuário requisitante (passo 11).

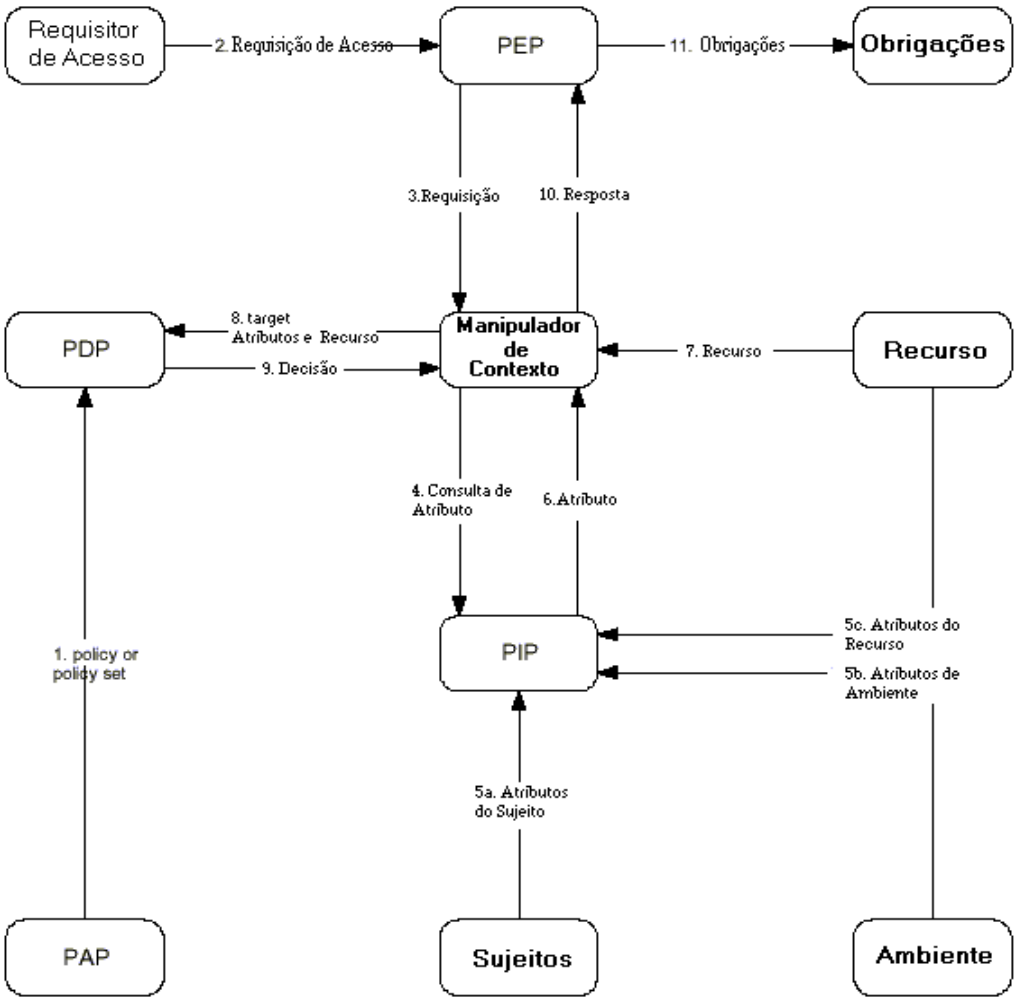

Figura 15 – Arquitetura de Autorização XACML Modelo Conceitual [GODIK 2003]

3.3.7. Estrutura de Políticas XACML

3.3.7.1. Policy e PolicySET

A estrutura raiz de todas as políticas XACML é formada por um elemento *Policy* ou um *PolicySet*. Um *PolicySet* é um elemento que pode possuir uma política ou um conjunto de políticas (*Policy*). O elemento *Policy* representa uma política de controle de acesso simples, expressa através de regras. A figura 16 demonstra um esquema da estrutura de políticas XACML.

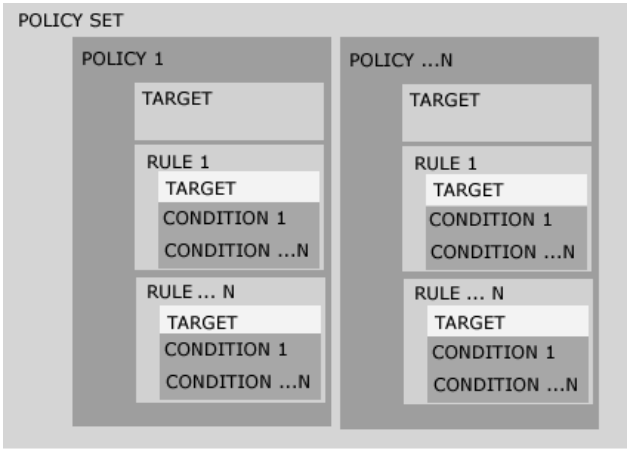

Figura 16 – Estrutura de Políticas XACML

Políticas XACML podem conter um único elemento raiz podendo ser um elemento *Policy* ou *PolicySet*. Tanto um elemento *Policy* como *PolicySet* podem conter um conjunto de regras e políticas, e cada conjunto pode corresponder a diferentes decisões de controle de acesso. Assim pode se tornar necessário o uso de ferramentas para facilitar a avaliação desses conjuntos de regras e políticas. Isso pode ser feito através do uso de Algoritmos de Combinação, que avaliam os grupos de regras e políticas combinando as múltiplas decisões em uma simples decisão. Os algoritmos de combinação são classificados em Algoritmos de Combinação de Políticas (*permit-overrides, deny-overrides, first-applicable, only-one-applicable-policy*) e Algoritmos de Combinação de Regras (*permit-overrides, deny-overrides, first-applicable*). Por exemplo, *Deny-Overrides Policy Algorithm* em sua definição verifica um conjunto de políticas, se alguma verificação retornar *Deny* ou não encontrar um *Permit*, o resultado final da combinação será *Deny*. Esses algoritmos de combinação estão previstos na

própria especificação do padrão XACML. A utilização de outros algoritmos de combinação é possível e suportado.

3.3.7.2. Target

O elemento *Target* é local onde são expressos um conjunto de condições que definem sujeitos, recursos e ações as quais a política ou regra se refere (figura 17). As definições contidas no *Target* são utilizadas para comparação com valores contidos em requisições de acesso, por exemplo, quando uma requisição solicita acesso a um determinado recurso, o valor que identifica um recurso deve estar expresso no *Target* da política que define o acesso a ele. O mesmo pode ocorrer com sujeitos. Para a comparação entre valores contidos no *Target* e em uma requisição são utilizadas funções lógicas. Quando todas as condições contidas no *Target* são satisfeitas as políticas ou regras são aplicadas. O *Target* pode ser utilizado também como indexadores de políticas, identificando políticas que se aplicam a sujeitos e objetos específicos.

```
06:  <Target>
07:    <Subjects>
08:      <AnySubject/>
09:    </Subjects>
10:    <Resources>
11:      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
12:        <AttributeValue
           DataType="http://www.w3.org/2001/XMLSchema#string">SampleServer</AttributeValue>
13:        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
14:                          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
15:      </ResourceMatch>
16:    </Resources>
17:    <Actions>
18:      <AnyAction/>
19:    </Actions>
20:  </Target>
```

Figura 17 – Exemplo de *Target*.

3.3.7.3. Rules

Uma política pode ser formada por varias regras de autorização (*rules*) que possuem informações lógicas que determinam quais os tipos de acesso que sujeitos possuem sobre objetos. Uma regra de autorização XACML pode ser basicamente formada por um *target* e uma condição (*Condition*), (figura 18). Uma condição é sempre avaliada por uma função lógica. Quando a condição é satisfeita retorna um efeito (*Deny* ou *Grant*). A avaliação de uma condição também pode retornar um erro (*Indeterminate*) e ainda pode retornar (*NotApplicate*), quando uma condição não se aplica a requisição de acesso.

```
46:   <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
47:     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal"
48:       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
49:         <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
50:                                       AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
51:       </Apply>
52:       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
53:     </Apply>
54:     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"
55:       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
56:         <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
57:                                       AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
58:       </Apply>
59:       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
60:     </Apply>
61:   </Condition>
```

Figura 18 – Exemplo de *Condition*.

A estrutura completa de uma política contendo regras de autorização pode ser visualizada na figura 19 e 20.

3.3.7.4. Atributos e Valores de Atributos

Atributos são características que servem para identificar particularidades de sujeitos, recursos, ações e ambientes onde requisições de acesso são criadas. Por exemplo, atributos podem ser utilizados para identificar sujeitos por seus nomes, sua habilitação, recursos que possui acesso e data e hora que os atributos são válidos.

Uma requisição de acesso possui atributos que são comparados a atributos de políticas para tomada de decisões. Para a resolução e comparação de atributos são utilizados dois mecanismos: o *AtributeDesignator* (figura 17) e o *AtributeSelector* (figura 18).

Um *AtributeDesignator* permite em uma política identificar um atributo e seu tipo de dados. São definidos quatro tipos de *AtributeDesignator*, para sujeitos (*Subject*), para recursos (*Resource*), para ações (*Action*) e para ambientes (*Environment*).

Um *AtributeSelector* serve para especificar um caminho para se obter um valor de atributo externo necessário para validar uma regra. Em um *AtributeSelector* o caminho é especificado através de uma expressão *XPath*. É necessário especificar o tipo de dados do valor do atributo indicado. Uma expressão *XPath* pode retornar um único valor ou um conjunto de valores quando uma expressão indica para um conjunto de Sub-elementos XML. A esse conjunto de valores é dado o nome *Bag*. É importante ressaltar que uma *Bag* só pode conter valores do mesmo tipo.

3.3.7.5. Funções

O XACML define o uso de funções em diferentes locais de uma política para manipulação de valores de atributos. O XACML utiliza funções para analisar um conjunto de valores contidos em *Bags* e compará-los com valores esperados em uma política. Funções podem trabalhar com qualquer combinação de valores de atributos e podem retornar qualquer tipo de atributos.

Funções podem ser combinadas para surtir um melhor efeito de coleta de valores. Do mesmo modo, novas funções podem ser criadas e utilizadas em conjunto com funções existentes.

O padrão XACML define uma coleção de funções padrões para tratar conjuntos de valores, e são chamadas de *Bag Functions*. Por exemplo, *Bag Functions* na forma *[type]-one-and-only*, acessam *Bags* de valores e retornam um valor específico e desejado, ou erro quando nenhum ou múltiplos valores são encontrados. Um exemplo do uso de funções em políticas XACML é apresentado na figura 20 nas linhas 47 e 48.

```
01: <Policy PolicyId="SamplePolicy"
02:       RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
03:   <Target>
04:     <Subjects>
05:       <AnySubject/>
06:     </Subjects>
07:     <Resources>
08:       <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
09:         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">SampleServer</AttributeValue>
10:         <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
11:                                      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
12:       </ResourceMatch>
13:     </Resources>
14:     <Actions>
15:       <AnyAction/>
16:     </Actions>
17:   </Target>
18:   <!-- Rule to see if we should allow the Subject to login -->
19:   <Rule RuleId="LoginRule" Effect="Permit">
20:     <!-- Only use this Rule if the action is login -->
21:     <Target>
22:       <Subjects>
23:         <AnySubject/>
24:       </Subjects>
25:       <Resources>
26:         <AnyResource/>
27:       </Resources>
28:       <Actions>
29:         <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
30:           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">login</AttributeValue>
31:           <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
32:                                      AttributeId="ServerAction"/>
33:         </ActionMatch>
34:       </Actions>
35:     </Target>
```

Figura 19 - Exemplo de uma Política XACML (parte 1).

```
36:
37:     <!-- Only allow logins from 9am to 5pm -->
38:
39:     <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
40:       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal"
41:         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
42:           <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
43:                                 AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
44:         </Apply>
45:         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
46:       </Apply>
47:       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"
48:         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
49:           <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
50:                                 AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
51:         </Apply>
52:         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
53:       </Apply>
54:     </Condition>
55:   </Rule>
56
57:   <!-- A final, "fall-through" Rule that always Denies -->
58:   <Rule RuleId="FinalRule" Effect="Deny"/>
60: </Policy>
```

Figura 20 - Exemplo de uma Política XACML (parte2).

3.4. CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou uma visão geral sobre o que são *Web Services* e como estes podem facilitar a disseminação de informações através da *web*. Também foram abordados os fundamentos á estrutura do padrão XML. O entendimento do padrão XML, de suas características e funcionalidades é de fundamental importância. Pelo fato do XML ser um padrão utilizado para a definição de todos os padrões para *Web Services*.

Este capítulo também apresentou as principais tecnologias e padrões XML utilizados para estabelecer segurança, para conteúdos digitais, através de *Web Services*. Abordando os padrões de segurança XML bem como suas características principais. Uso desses padrões de segurança XML permitem que *Web Services* transfiram informações através da *web*. A facilidade oferecida pelo uso desses padrões se estende por diferentes domínios *web*, possibilitando uma integração segura entre diferentes *Web Services* e seus usuários.

4. TRABALHOS RELACIONADOS

Vários esforços têm sido feitos nos últimos anos para definir protocolos e padrões para a segurança de *Web Services*. Os *Web Services* são baseados em XML e necessitam de implementação de segurança, pois inicialmente os primeiros protocolos para troca de mensagens, como o SOAP, não foram projetados para possuir características de segurança.

Soluções proprietárias para implementação de segurança em *Web Services* são precárias, no sentido de não promover interoperabilidade entre diferentes plataformas e domínios *web* [DEITEL 2002].

A troca de mensagens entre entidades, de maneira segura, é o primeiro passo para a obtenção de segurança em *Web Services*. Através da troca de mensagens pode-se, por exemplo, autenticar entidades e autorizar a utilização de recursos.

Vários trabalhos foram desenvolvidos buscando a definição de linguagens que permitissem a troca de mensagens seguras entre entidades *web* [EASTLAKE 2002a] [EASTLAKE 2002b]. Os padrões XML-Signature e XMLEncryption fazem parte dos esforços para garantir a autenticidade e integridade na troca de mensagens seguras para *Web Services*.

A disponibilidade de recursos depende de mecanismos de segurança que possam realizar a autenticação e autorização de entidades requisitantes de *Web Services*. A autorização de entidades e distribuição de direitos sobre recursos pode ser efetuada através do uso de padrões como XACML e SAML. O fluxo de troca de mensagens, responsáveis por uma autorização, é determinado através de uma arquitetura autorização. Uma arquitetura de autorização determina quais as entidades são responsáveis por executar o controle de acesso aos recursos bem como quais são os mecanismos utilizados para o controle de acesso.

São apresentados nessa seção trabalhos realizados diretamente relacionados à arquitetura de autorização e controle de acesso a conteúdos digitais.

4.1. ARQUITETURA DE UM PROCESSADOR DE CONTROLE DE ACESSO DISTRIBUÍDO PARA WEB

Reiner Kraft em [KRAFT 2002] apresenta em seu trabalho uma proposta para uma arquitetura de processador de controle de acesso distribuído para *Web Services*. A arquitetura do processador de controle de acesso é baseada em um modelo abstrato de serviços de rede na *web*.

4.1.1. Modelo Abstrato de Serviços de Rede na Web

O modelo de redes de *Web Services* define componentes de uma Arquitetura Orientada a Serviços (*Service Oriented Architecture* - SOA). Os componentes definidos são objetos de serviços *web* (*web services objects*), métodos de serviços *web* (*web services methods*) e coleções de serviços *web* (*web services collections)*.

Kraft utiliza uma notação de programação orientada a objetos para a definição de objetos de serviços *web* onde o objeto encapsula dados e comportamentos. Um objeto de serviço *web* é o componente o qual aplicações podem interagir através de trocas de mensagens. O objeto de serviço *web* é definido através de uma tupla que identifica seus sete elementos internos (figura 21): a URN; o protocolo de troca de mensagens (SOAP); a URL; um conjunto de estados; as operações definidas nos métodos de serviços *web*; a coleção de serviços *web* que o objeto pertence; um conjunto de metadados que informações adicionais sobre o objeto; e um conjunto finito de processadores da controle de acesso (ACPs) que efetuam a decisão de autorização sobre o objeto.

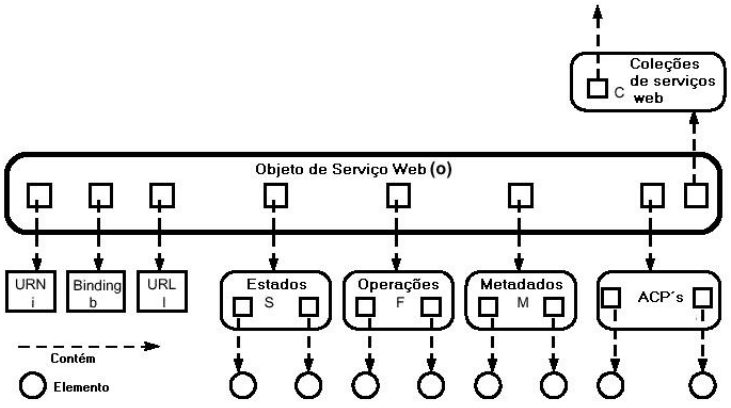

Figura 21 – Web Service Object [KRAFT 2002].

O método de serviço *web* representa uma operação em um objeto de serviço *web*, e pode também ser visto como um recurso.

O método de serviço *web* é definido através de uma tupla de cinco elementos (figura 22) que determinam: a URN; o objeto qual pertence o método; um conjunto de parâmetros de entrada; a função de execução; e um conjunto de metadados para informações sobre o método.

Uma coleção de serviços *web* representa a união ou grupo de objetos de serviços *web* e é definida através de uma tupla composta de seis elementos que podem definir (figura 22): a URN; um conjunto de objetos; um conjunto de outras coleções de serviços *web* filhos; uma coleção de *serviços web* pai; um conjunto metadados para informações adicionais; e um conjunto finito de processadores de controle de acesso.

Reiner Kraft ainda demonstra como combinar simples *Web Services* com outros para formar um novo *Web Service*.

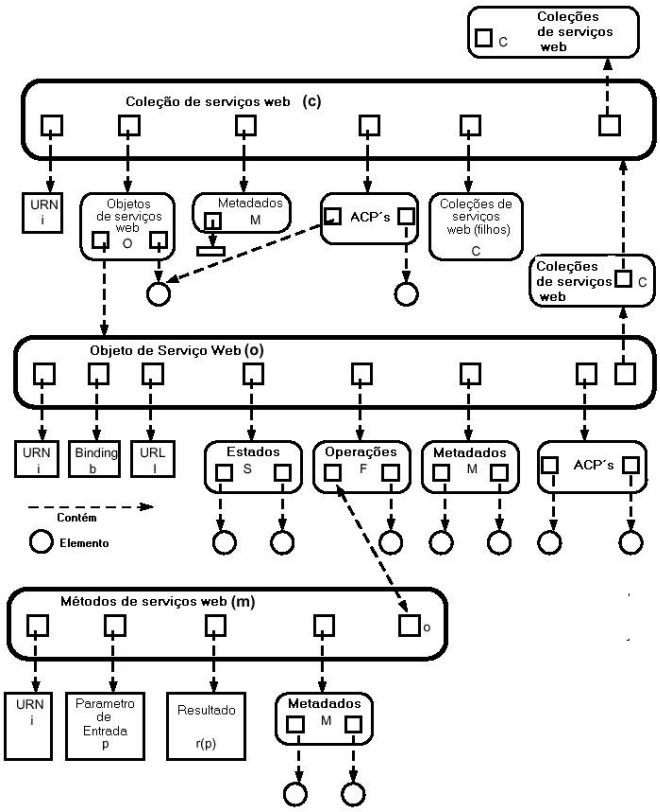

Figura 22 – Web Service Collection , Method e Object [KRAFT 2002].

4.1.2. Processador de Controle de Acesso Distribuído

Reiner Kraft apresenta um modelo para controle de acesso distribuído para *Web Services* descrevendo os componentes formadores da arquitetura proposta. Os

componentes são o processador de controle de acesso (*Access Control Processor*) e o *GateKeeper*.

O processador de controle de acesso é um objeto de serviços *web* que faz a tomada de decisões de autorização sobre um componente de *Web Services*. Um processador de controle de acesso pode tomar decisões em conjunto com outros processadores de controle de acesso. Um componente de *Web Services* pode ter vários processadores de controle de acesso associados a ele.

O *GateKeeper* é um processador de controle de acesso que faz a tomada de decisão final sobre o acesso a um componente de *Web Services*. Ele também pode autenticar sujeitos e gerar identidades para essa autenticação. Um componente de *Web Services* pode possuir apenas um *GateKeeper* a associado a ele.

A figura 23 demonstra um exemplo que ajuda a mostrar a arquitetura de processador de controle de acesso distribuído. Nessa arquitetura um cliente deseja acessar um objeto de serviço web (3) e a requisição é interceptada por um *GateKeeper* (ACP 1) que autentica o cliente e redireciona a requisição para o processador de controle de acesso (ACP 2). O ACP1 deve comunicar-se com outros ACPs para recolher todas as decisões de acesso referentes a um recurso. Depois o *GateKeeper* coleta as informações de autorização que ele usa para tomar a decisão de acesso. Finalmente o *GateKeeper* envia uma resposta contendo a decisão de autorização ao cliente.

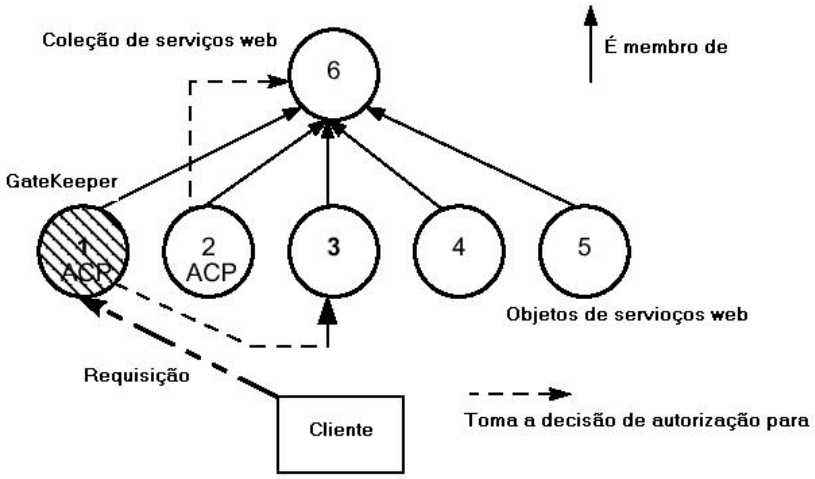

Figura 23 – Cenário de processador para controle de acesso distribuído para *web services* [KRAFT 2002].

Nesse trabalho Kraft define tuplas que formam o Processador de Controle de Acesso e o *GateKeeper*. Kraft também define um algoritmo para determinar a responsabilidade de tomada de decisão de autorização. Esse algoritmo é utilizado para determinar quais componentes são responsáveis pelas decisões de acesso de um determinado recurso. As trocas de mensagens de autorização são feitas utilizando o protocolo SOAP, mas o autor deixa claro que a utilização de padrões como SAML podem ser utilizados para melhorar a arquitetura proposta. Kraft não define como autorizações são definidas e armazenadas em sua proposta, mas sugere o uso do padrão XACML como uma das alternativas para esse caso.

4.2. LINGUAGENS DE CONTROLE DE ACESSO A DOCUMENTOS XML

Vários trabalhos de pesquisa foram realizados no sentido de promover controle de acesso em documentos XML, entre eles pode-se destacar [HADA 2000b], [DAMIANI 2002] e [GABILON 2001]. Kudo e Hada em seu trabalho [HADA 2000b] definiram uma arquitetura de autorização e uma linguagem de definição de políticas de autorização. Na proposta [HADA 2000b] são definidos dois módulos diferentes, chamados *Provisional Autorization Module* (PAM) e *Request Execution Module* (REM), utilizados para a autorização e execução de requisições, respectivamente. A decisão de autorização é feita através do módulo PAM utilizando para isso regras de autorização e políticas de segurança implementadas através da linguagem de controle de acesso XACL definida na proposta. A figura 24 apresenta a um esquema da arquitetura proposta por Kudo e Hada.

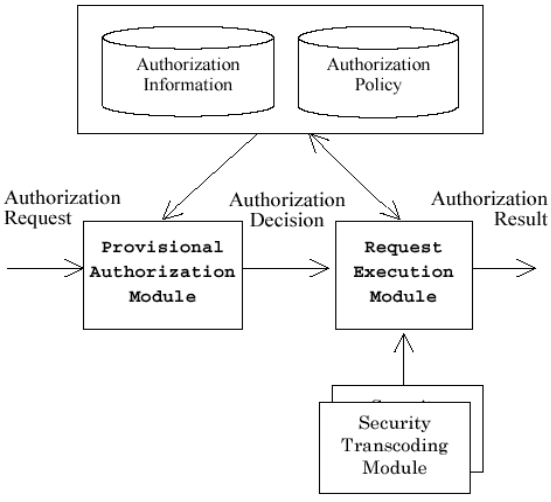

Figura 24 – Provisional Authorization Architecture [HADA 2000b].

Em [DAMINANI 2002] é apresentada uma arquitetura de controle de acesso para documentos XML utilizando expressões XPath e XPointer dentro de políticas para se referir a asserções sobre recursos e sujeitos, definindo uma linguagem para criação de regras de autorização.

Em [GABILON 2001] apresenta-se também uma arquitetura de autorização para controle de acesso em documentos XML onde sujeitos são armazenados em um documento XML que obedece à hierarquia de usuários definida no documento *XML Subject Sheet* (XSS) e também são utilizadas expressões XPath para identificar usuários. Nessa proposta também é definida uma linguagem para definição de regras de autorização bem como o algoritmo que permite coletar as regras referentes a cada sujeito e recurso.

É importante ressaltar que todos esses trabalhos se preocupam em definir uma linguagem de especificação de políticas de controle de acesso baseadas em XML e como proceder para recuperar essas políticas para uma posterior avaliação. O processo de avaliação dessas políticas que resulta na tomada de decisão não é abordada nesses trabalhos.

4.3. MODELO DE CONTROLE DE USO ATRAVÉS DE CONTROLE DE ACESSO TRADICIONAL

Park e Sandhu em [PARK 2002] desenvolvem em seu trabalho um conceito de controle de uso (*Usage Control* – UCON) envolvendo controle de acesso tradicional, gerenciamento de confiança e gerenciamento de direitos digitais. Eles definem também os componentes do UCON e como autorização e controle de acesso podem ser aplicados ao modelo especificado. A figura 25 apresenta o escopo para o controle de uso.

O modelo apresentado define um controle de domínio como área de cobertura, onde direitos e o uso dos direitos sobre conteúdos digitais são controlados por um monitor de referência. São definidos dois tipos de controle de domínios baseados na localização do monitor de referencia.

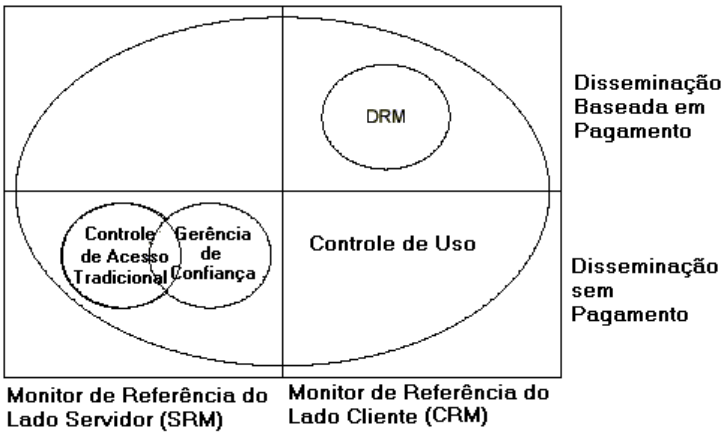

Figura 25 – Escopo de Controle de uso [PARK 2002].

O primeiro é o controle de domínio com monitor de referência do lado servidor (SRM). E o segundo é o controle de domínio com monitor de referência no lado cliente (CRM). O SRM (figura 26.a) reside no lado servidor e faz a mediação de todos os acessos a conteúdos digitais. O CRM (figura 26.b) reside no lado cliente e controla o acesso e o uso de conteúdos digitais.

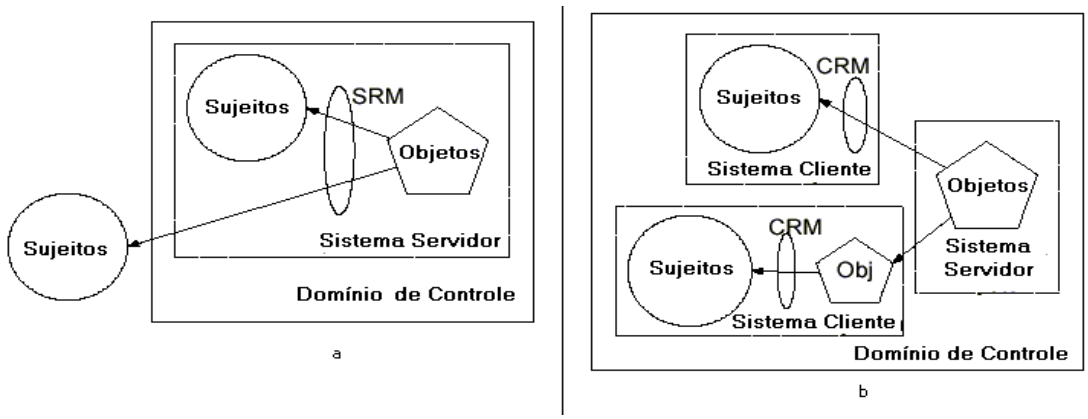

Figura 26 – Modelo de Controle de Domínio SRM(a) e CRM(b) [PARK 2002].

O modelo de controle de uso proposto consiste em três componentes principais e três componentes adicionais que são fortemente ligados ao processo de autorização (figura 27). Os componentes principais compreendem sujeitos, objetos e direitos. Os componentes adicionais são regras de autorização, condições e obrigações.

No modelo UCON as regras de autorização, condições e obrigações são utilizadas no processo de autorização combinando entre si e com outros componentes, como

direitos (*rights*), para definir 4 tipos de autorizações suportando os mecanismos como o MAC, DAC, RBAC e DRM.

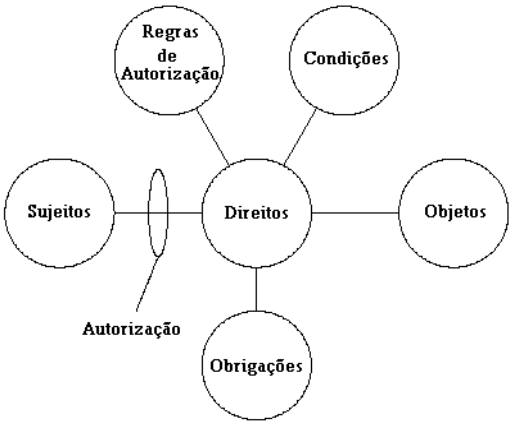

Figura 27 – Componentes do modelo UCON.

4.4. CONCLUSÃO DO CAPÍTULO

Os trabalhos apresentados desenvolvem idéias diferentes para controle de acesso a conteúdos digitais, propondo arquiteturas e uso de novos padrões de segurança para facilitar o acesso a informações sensíveis. Todos os trabalhos sugerem a proteção e distribuição de conteúdos digitais, no sentido de fornecer uma base segura para o processo de autorização.

A complexidade para definição de uma infra-estrutura completa para segurança em *Web Services* pode ser medida através da capacidade de expansão das necessidades de segurança, que podem surgir em um ambiente distribuído como a *web*, devido ao grande poder de flexibilidade que a *web* pode apresentar.

O uso de padrões de segurança abertos pode melhorar o aspecto de definição de uma infra-estrutura de controle de acesso e autorização, na medida que estabelecem protocolos de comunicação e trocas de mensagens que fornecem a interoperabilidade entre multi-plataformas e domínios bem como a definição de linguagens para especificação de políticas de controle de acesso e autorização.

O uso de padrões como XACML e SAML são abordados apenas de maneira superficial nesses trabalhos. Não havendo assim um aprofundamento no sentido de como esses padrões podem facilitar o processo de autorização de entidades e distribuição de direitos sobre recursos.

5. MODELO DE AUTORIZAÇÃO E DISTRIBUIÇÃO DE DIREITOS DE ACESSO SOBRE CONTEÚDOS DIGITAIS

Esta dissertação apresenta uma proposta para modelo para autorização e distribuição de direitos de acesso para conteúdos digitais aplicando novos paradigmas de segurança em *Web Services*. O modelo prevê um processo de avaliação[1] de políticas de controle de acesso para execução do processo de autorização. O modelo de distribuição de direitos de acesso também apresenta as diferentes entidades responsáveis pelo processo de avaliação de políticas de segurança e como elas podem dar suporte a um serviço de distribuição de licenças de uso.

O modelo de autorização propõe o uso de padrões de segurança baseados em XML, como a XACML (*eXtensible Access Control Markup Language*), na definição de políticas de autorização e controle de acesso. Para a distribuição desses direitos de acesso é proposto o uso do padrão SAML (*Security Assertion Markup Language*), que é usado para troca de mensagens entre entidades do sistema atuando como uma linguagem de distribuição de direitos. Os direitos são definidos com base nas políticas de segurança que são utilizadas para determinar as decisões de acesso sobre os conteúdos digitais, mediante uma requisição de autorização.

Posteriormente os direitos de acesso disponibilizados por esse modelo podem ser utilizados, por exemplo, para a construção de licenças DRM[2] (*Digital Rigths*

---

[1] O termo *avaliação de políticas* está ligado ao processo de decisão, onde as regras de autorização de uma política são *avaliadas* para gerar uma decisão de autorização.

[2] Digital Rights Management (DRM) é um conjunto de softwares, serviços e tecnologias que limitam e controlam o uso de conteúdo digital e gerenciam conseqüências deste uso durante todo o ciclo de vida do conteúdo. DRM é uma tentativa de possibilitar proteção persistente de conteúdo,ou seja, proteção que permanece com o conteúdo depois deste ser disponibilizado [STAMP 2002].

Management) [CHONG 2002] [FILIPPIN 2004] e também para um controle de uso de conteúdos digitais.

5.1. MODELO GERAL DE DISTRIBUIÇÃO DE LICENÇAS

Para aplicar a estrutura de autorização para distribuição de direitos de acesso em uma arquitetura de distribuição de licenças é necessário definir como as entidades podem interagir no sistema.

O modelo geral para distribuição de licenças [FILIPPIN 2004] apresentado na figura 28 é formado pelas entidades definidas como: *Entidade de Autenticação* (EA); *Entidade de Conteúdo* (EC); *Entidade de Criação, Distribuição e Gerenciamento de Licenças* (ECDGL); e *Entidade de Autorização e Distribuição de Direitos de Acesso* (EADD).

A *Entidade de Autenticação* (EA), é responsável por recolher as informações necessárias para identificar um usuário, e disponibilizá-las para os outros componentes do sistema, funcionando como uma base de informações de atributos de usuários.

A *Entidade de Conteúdo* (EC), é responsável por distribuir conteúdos protegidos aos usuários autenticados no sistema. Esta serve também como base de informações de atributos de conteúdos protegidos. A entidade de conteúdo de também conter informações sobre o processo de pagamento e assinaturas que podem determinar a validade de licenças de uso.

A *Entidade de Criação, Distribuição e Gerenciamento de Licenças* (ECDGL) é responsável por criar e distribuir licenças necessárias para o controle de uso de conteúdos protegidos. O processo de geração de licenças é feito com base nos direitos de acesso que o usuário possui sobre os conteúdos digitais protegidos. Esses direitos são determinados através da avaliação de políticas de controle de acesso durante o processo de autorização.

A *Entidade de Autorização e Distribuição de Direitos de Acesso* (EADD) é responsável por avaliar políticas de controle de acesso e determinar quais os direitos de acesso que o usuário possui sobre um conteúdo digital protegido específico, e disponibilizar esses direitos para a entidade ECDGL.

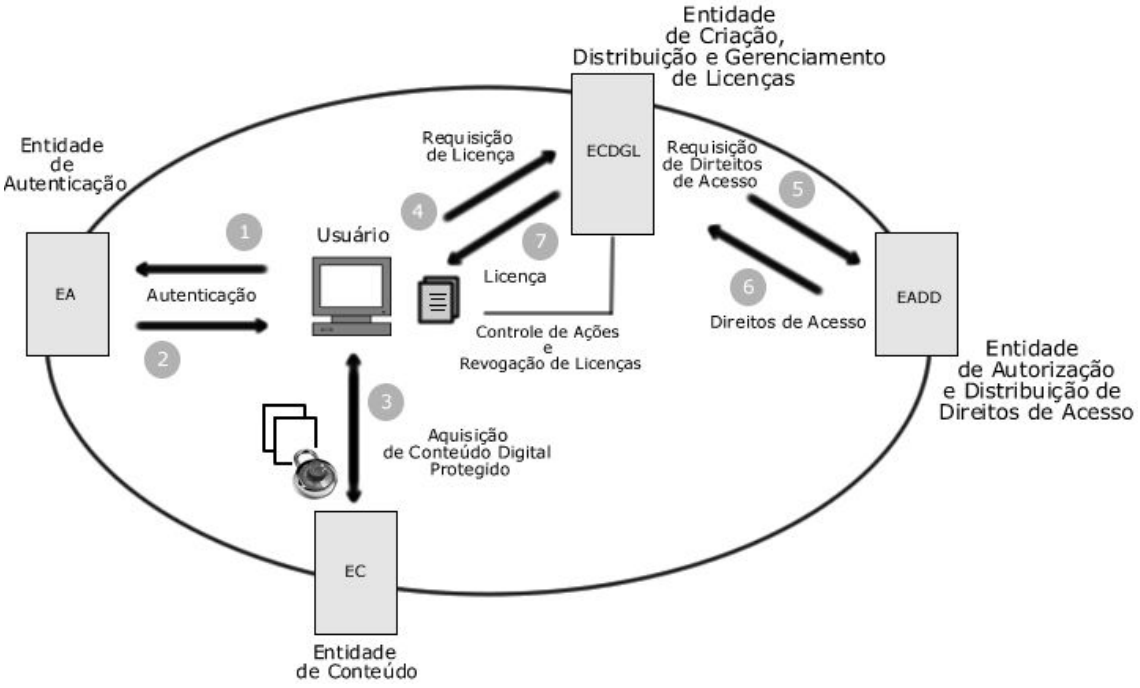

Figura 28 - Modelo de Distribuição de Licenças.

A figura 28 apresenta o fluxo de trocas de informações que ocorrem quando o usuário cliente requisita uma licença de uso para um conteúdo adquirido. Primeiro, o usuário é autenticado pelo servidor de autenticação (passos 1 e 2). Depois, o usuário faz a requisição de licença de uso (passo 4) para um conteúdo específico adquirido (passo 3). É importante observar que o usuário é definido como uma aplicação de controle de conteúdo que é executada no lado cliente. Então, o servidor de licenças formula e envia uma requisição de autorização de direitos ao servidor de autorização (passo 5). O servidor de autorização efetua o processo de autorização e envia uma mensagem contendo a decisão de autorização sobre os direitos requeridos pelo usuário (passo 6). Finalmente o servidor de licenças monta a licença e remete ao usuário (passo 7).

O modelo autorização proposto nesta dissertação se aprofunda no modelo geral de distribuição de licenças (figura 28), concentrando-se na definição da estrutura e funcionamento da entidade de autorização e distribuição de direitos de acesso.

5.2. MODELO DISTRIBUÍDO PARA TROCA DE MENSAGENS ENTRE ENTIDADES PARA AUTORIZAÇÃO

Em um ambiente *web* geralmente as entidades que participam do processo de autorização podem estar distribuídas na rede. Identificar e definir as entidades que participam do processo de autorização é necessário para se definir o papel que cada uma delas pode e deve exercer. Assim torna-se necessário o uso de protocolos que permitam que essas entidades possam comunicar-se umas com as outras para efetuar o processo de autorização.

O modelo apresentado na figura 29 demonstra uma visão global das entidades envolvidas no processo de autorização e onde o uso dos padrões SAML e XACML pode ser feito para efetuar as trocas de mensagens de requisição e resposta para possibilitar o processo de autorização. O modelo define as entidades que podem estar envolvidas em um processo de autorização baseado na arquitetura de autorização AAA *Authorization Framework* da IETF [RFC-2904 1998], e também no modelo conceitual abordado na própria especificação do padrão XACML [GODIK 2003].

As entidades que podem estar envolvidas são o PEP, PDP, PIP, PAP e o PRP. A maioria dessas entidades possui as mesmas funcionalidades apresentadas no modelo XACML na seção 3.3.5 deste trabalho. Neste modelo o PAP é diferente, pois nesse contexto assume apenas a função de criação e armazenamento de políticas e o PRP (*Policy Retrieval Point*) torna-se responsável por recuperar as políticas necessárias para o processo de autorização. Já o PIP é responsável por disponibilizar e também recuperar qualquer atributo.

A seqüência de trocas de mensagens para efetuar o processo de autorização segue os seguintes passos:

1. Um usuário requisita acesso a uma informação protegida;
2. O PEP intercepta a requisição e encaminha uma requisição de acesso SAML para uma aplicação *middleware* que faz o papel de manipulador de contexto;
3. O manipulador de contexto pode requisitar atributos extras ao PIP, necessários para a decisão de autorização, isso pode ser feito usando SAML;

4. O PIP recupera os atributos necessários requisitados pelo manipulador de contexto;
5. O PIP envia uma asserção de atributos SAML ao manipulador de contexto;
6. O Manipulador de contexto monta a requisição de decisão de autorização em XACML e envia para o PDP;
7. O PDP requisita as políticas necessárias ao PRP;
8. O PRP recupera as políticas referentes ao usuário e o conteúdo;
9. O PRP envia as políticas para o PDP;
10. O PDP avalia as políticas e retorna uma decisão de autorização XACML;
11. O Manipulador de contexto envia ao PEP uma asserção de acesso SAML;
12. Caso seja permitido o PEP aplica a decisão sobre o usuário e conteúdo;
13. Caso acesso seja negado o PEP responde ao usuário;

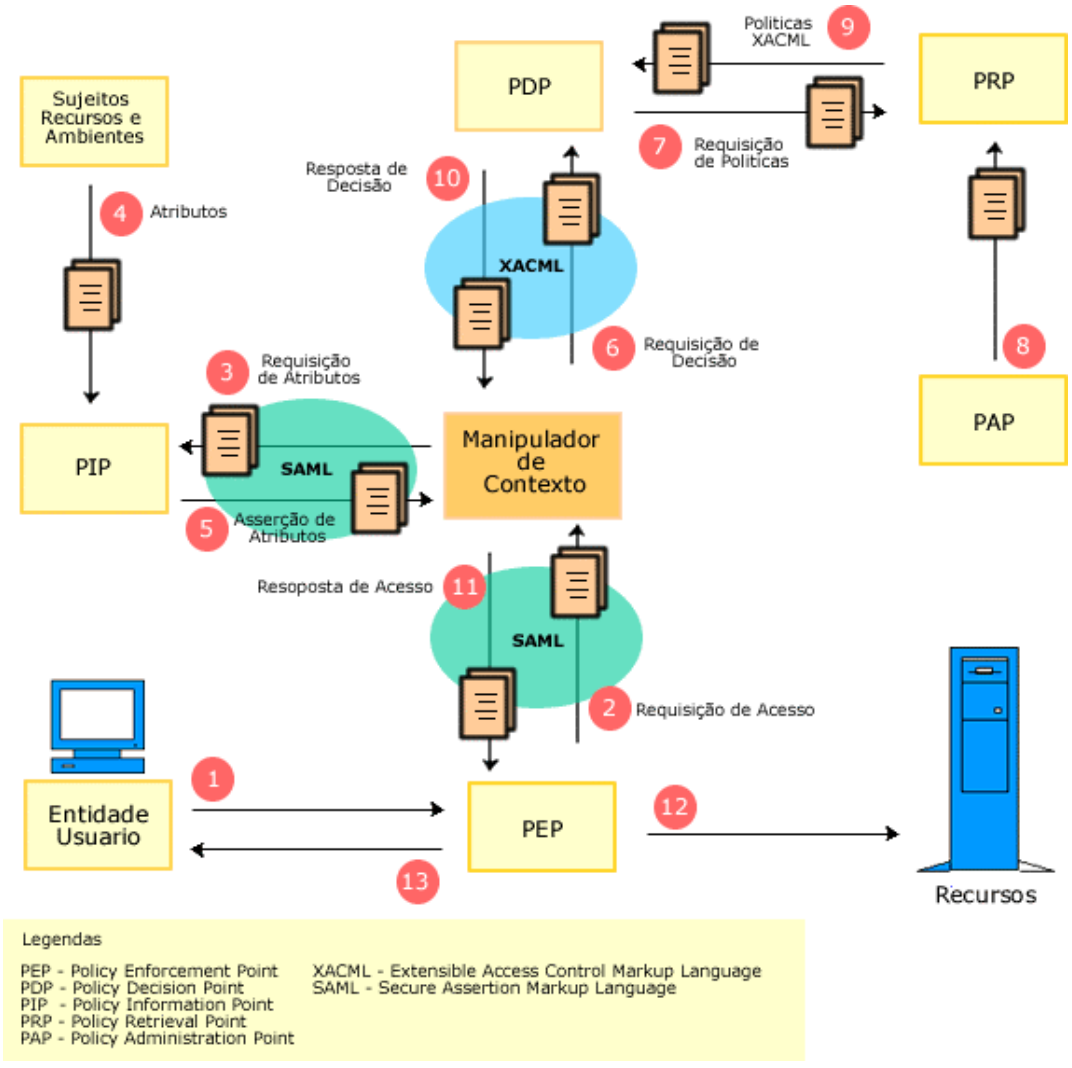

Figura 29 – Fluxo de troca de mensagens entre entidades para autorização.

O uso dos padrões de segurança XML é muito importante para estabelecer uma troca de informações segura entre entidades distribuídas. O SAML e XACML não se preocupam com integridade e autenticidade de mensagens, mas apenas com o vocabulário para estabelecer um padrão de asserções de requisição e decisão de acesso. Para reforçar a segurança na troca de mensagens entre as entidades pode-se utilizar o XML *Encryption* e XML-*Signature*, descritos na seção 3.3.2 e 3.3.1 deste trabalho. Assim, por exemplo, uma asserção SAML pode ser cifrada utilizando XML *Encryption* e assinada utilizando o XML-*Signature*. O conteúdo gerado nesse processo pode ser encapsulado no corpo de um envelope de mensagem SOAP e enviado para seu destino. A figura 30 mostra o formato de um envelope SOAP e onde os padrões de segurança XML podem ser aplicados.

Figura 30 – Envelope SOAP

5.3. MODELO DE AUTORIZAÇÃO E DISTRIBUIÇÃO DE DIREITOS DE ACESSO

O processo de autorização implica em um processo de tomada de decisão. Geralmente o processo de tomada decisão é executado através da avaliação de políticas de controle de acesso sobre um sujeito ou objeto, este processo determina os direitos que um sujeito tem sobre um determinado objeto.

O modelo de autorização sobre direitos de acesso define as entidades responsáveis por disponibilizar as políticas e atributos necessários para o processo de decisão, bem como, quais são as entidades diretamente responsáveis pelo processo de decisão. O modelo também define como ocorrem as trocas de mensagens para a requisição e distribuição de direitos de acesso entre uma entidade gerenciadora de licenças (ECDGL) e uma entidade de autorização (EADD).

O modelo de autorização sobre direitos de acesso pode ser observado na figura 31 onde as entidades de autorização aparecem distribuídas em diferentes ambientes.

Esse modelo pode ser comparado com o modelo de trocas de mensagens (seção 5.2), onde a entidade ECDGL assume o papel do PEP, interceptando uma requisição de licença do usuário e encaminhando uma requisição de acesso a entidade EADD. A requisição de licença do usuário pode conter informações que identificam o usuário, por exemplo, uma chave pública do usuário, e os direitos que o usuário está requisitando, por exemplo, visualizar um documento que foi adquirido.

A requisição de acesso nesse modelo é uma requisição de direitos de acesso requeridos pelo usuário, que é enviada da entidade ECDGL à entidade EADD.

A entidade EADD modelo assume as funcionalidades de PDP (item 6), PRP (item 7) e PIP (item 8). Essas funcionalidades são englobadas em uma aplicação de manipulação de contexto definida como Aplicação de Autorização e Distribuição de Direitos de Acesso (AADD). É importante ressaltar que outras entidades do modelo também podem englobar características de recuperação e distribuição de atributos através de um PIP.

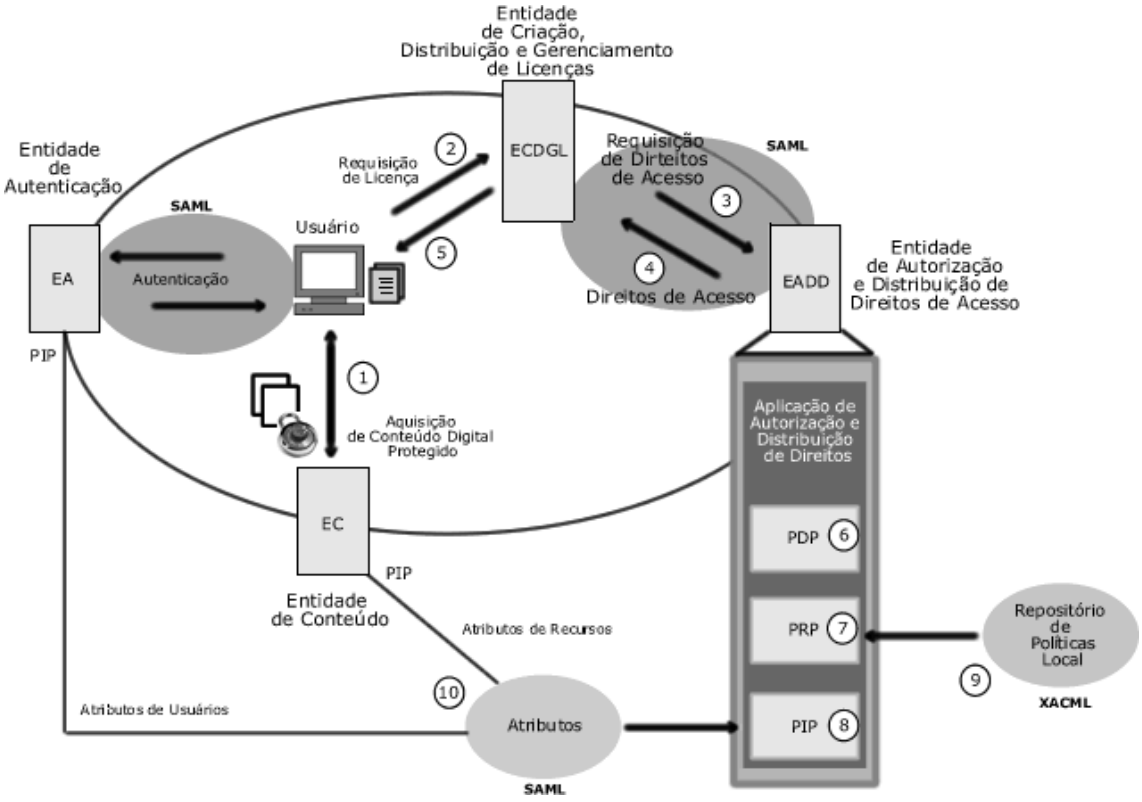

Figura 31 - Modelo de autorização sobre direitos de acesso aplicado ao modelo geral.

A aplicação AADD se encarrega de recuperar políticas, recuperar e distribuir atributos e avaliar políticas executando o processo de decisão. O resultado desse processo é a resposta contendo os direitos requeridos sobre um recurso (passo 4).

A vantagem da utilização de um modelo distribuído está na organização e divisão de tarefas entre suas entidades. A escalabilidade também é um fator importante e pode ser observado também como vantagem em um modelo distribuído, pelo fato de que a aplicabilidade do modelo não se altera independentemente da quantidade de objetos envolvidos no ambiente.

5.3.1. Definição de Políticas de Controle de Acesso

As políticas de controle de acesso são utilizadas no processo de autorização para definir decisão de acesso a conteúdos digitais protegidos. A definição das políticas controle de acesso podem ser baseadas nos modelos de controle de acesso MAC, DAC e RBAC. A escolha de um desses modelos é feita com base na necessidade do ambiente onde as políticas serão aplicadas.

As políticas de controle de acesso são implementadas através do uso padrão XACML, que proporciona um vocabulário rico e flexível que permite a implementação de qualquer tipo de política de controle de acesso. A partir do uso do XACML é possível definir políticas individuais, ou seja, políticas que se aplicam a sujeitos, objetos e ações específicas. Com o uso do XACML também é possível definir políticas globais, por exemplo, políticas que se aplicam a todos os recursos de um sistema.

```
<Policy  policyId="SamplePolicy"
      RuleCombiningAlgId="combining-algorithm:permit-overrides">
 <Description> Esta politica refere-se aos acessos de visualização de qualquer  usuarios a qualquer conteúdo  </Description>

 <Target>
    <Subjects>
           <Anysubject/>
   </Subjects>
   <Resources xmlns rec:="http://localhost/recursos">
          < AnyResource/>
   </Resources>
   <Actions>
      <ActionMatch MatchId="string-equal">
      <AttributeValue DataType="#string">Visualizar</AttributeValue>
      <ActionAttributeDesignator DataType="#string"
                    AttributeId="Acao"/> </ ActionMatch >
   </Actions>
 </Target>
```

Figura 32 – Exemplo de Política de Controle de Acesso XACML (parte 1).

```
  <Rule RuleId="1" Effect="Permit">
    <Target>
     <Subjects>
              <Subject xmlns user:="http://localhost/usuarios">
                       <SubjectMatch MatchId="Local:function:regexp-string-match">
                         <SubjectAttributeDesignator AttributeId="Classe-id" DataType="#string"/>
                         <AttributeValue DataType="#string">A</AttributeValue>
                 </SubjectMatch></Subject>
     </Subjects>
     <Resources>
                 <AnyResource/>
     </Resources>
     <Actions>
              <ActionMatch MatchId="string-equal">
                       <AttributeValue DataType="#string">Visualizar</AttributeValue>
                       <ActionAttributeDesignator DataType="#string"
                                                  AttributeId="Acao"/>
              </ActionMatch>
     </Actions>
    </Target>
  </Rule>

<Rule RuleId="2" Effect="Permit">
    <Target>
     <Subjects>
      <Subject xmlns user:="http://localhost/usuarios">
                       <SubjectMatch MatchId="Local:function:regexp-string-match">
                              <SubjectAttributeDesignator AttributeId="Classe-id" DataType="#string"/>
                              <AttributeValue DataType="#string">B</AttributeValue>
                       </SubjectMatch></Subject>
     </Subjects>
     <Resources>
                 <AnyResource/>
     </Resources>
     <Actions>
                 <ActionMatch MatchId="string-equal">
                     <AttributeValue DataType="#string">Visualizar</AttributeValue>
                     <ActionAttributeDesignator DataType="#string"
                                                AttributeId="Acao"/>
                 </ActionMatch>
     </Actions>
  </Target>
  <Condition>
            <Apply FunctionId="function:Date-less-than-or-equal"
                       <Apply FunctionId="date-one-and-only">
                        <ResourceAttributeDesignator AttributeId="DataPublicação"
                                             DataType="#date"/>
                       </Apply>
             <AttributeValue DataType="#Date">01/01/2002</AttributeValue>
           </Apply>
    </Condition>
 </Rule>
</Policy>
```

Figura 33 – Exemplo de Política de Controle de Acesso XACML (parte 2).

As figuras 32 e 33 demonstram um exemplo de política de controle de acesso XACML para distribuição de direitos de visualização de *E-Books*, que determina os direitos de visualização sobre qualquer conteúdo digital protegido. Nesse exemplo a primeira regra determina que usuários da classe “A”, são permitidos a visualizar qualquer conteúdo digital protegido. Na segunda regra é determinado que aos usuários

da classe “B” é permitido visualizar qualquer conteúdo digital protegido, desde que a data de publicação do conteúdo seja inferior a data de 01/01/2002.

A vantagem de se utilizar um padrão para a definição de políticas de controle de acesso é a possibilidade de interoperabilidade entre entidades de autorização em domínios diferentes. Isso pode ser visualizado na proposta de Kraft [KRAFT 2002] sobre um controle de acesso distribuído, onde um ACP (*Access Control Processor*) comunica-se com outros ACPs para recolher todas as decisões de acesso sobre um recurso, e depois efetuar a decisão final de acesso.

Os mesmos conceitos apresentados por Kraft [KRAFT 2002] podem ser utilizados nesse modelo. A distribuição do controle de acesso pode ser feita através do compartilhamento de políticas de controle de acesso entre diferentes domínios. Por exemplo, um servidor de autorização pertencente a um domínio “A” precisa avaliar políticas para definir os direitos de um usuário que pertence ao domínio “B”. Assim o servidor “A” precisa avaliar políticas que foram implementadas no domínio “B”, para efetuar um processo de tomada de decisão.

5.3.2. Requisição e Distribuição de Direitos

A troca de mensagens, que determina o processo de requisição e distribuição de direitos, pode ser realizada através do uso do padrão SAML. A requisição de direitos é emitida pelo servidor de licenças, com destino ao servidor de autorização. Essa requisição de direitos é formatada seguindo o formato de uma requisição de autorização que é especificada no padrão SAML. A figura 34 apresenta um exemplo de requisição de autorização SAML, onde o servidor de licenças solicita a autorização para “leitura” sobre o conteúdo “Livro.pdf” para o usuário “Joao”.

```
<Request IssueInstant="2003-04-17T00:46:02Z"
            MajorVersion="1"
            MinorVersion="1"
            RequestID="_c7055387-af61-4fce-8b98-e2927324b306">

  <AuthorizationDecisionQuery Resource="livro.Pdf">
        <Subject>
              <NameIndentifier SecurityDomain="Company.com"
               Name="Joao"/>
        </Subject>
        <Actions>
              <Action>Read</Action>
        <Actions>
  </AuthorizationQuery>
</Request>
```

Figura 34 – Exemplo de Requisição de Autorização SAML.

O servidor de autorização, após receber a requisição apresentada na figura 34, executa o processo de autorização e responde ao servidor de licenças com uma asserção de decisão de autorização SAML, encasulada em um objeto *Response* SAML. A figura 35 apresenta um exemplo de resposta emitida pelo servidor de autorização contendo uma asserção SAML que determina que o usuário "joao" possui direito de ler o documento "Livro.pdf".

```
<Response IssueInstant="2003-04-17T00:46:02Z"
            MajorVersion="1"
             MinorVersion="1"
             Recipient="www.servidorautorizacao.com"
             ResponseID="_c7055387-af61-4fce-8b98-e2927324b306">

 <Status> <StatusCode Value="Success"/> </Status>

 <Assertion AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
       IssueInstant="2003-04-17T00:46:02Z"
       Issuer="www.servidorautorizacao.com"
       MajorVersion="1"
       MinorVersion="1">

  <Conditions
    NotBefore="2003-04-17T00:46:02Z"
    NotOnOrAfter="2003-04-17T00:51:02Z">
  </Conditions>
    <AuthorizationDecisionStatement
                        Decision="Permit"
                        Resource="livro.Pdf">
            <Actions>
                        <Action> Read <Action>
            <Actions>
            <Subject>
                        <NameIdentifier
                                    SecurityDomain="Company.com"
                                    Name="joao"/>
            </Subject>
  </AuthorizationDecisionStatement>
 </Assertion>
</Response>
```

Figura 35 – Exemplo de Resposta de Autorização SAML.

No SAML os emissores de requisições e respostas são identificados através do uso do padrão XML-*Signature* através do elemento *Signature*. Assim é possível estabelecer confiança entre os servidores durante a troca mensagens. Por exemplo, é possível usar certificados SPKI para determinar a confiança na troca de mensagens entre o servidor de licenças e o servidor de autorização.

5.4. PROCESSO DE AUTORIZAÇÃO

O processo de autorização determina os direitos que um usuário possui sobre um conteúdo digital protegido. Park e Sandhu em [PARK 2002] definem um modelo onde

direitos e o uso dos direitos sobre conteúdos digitais são controlados por um monitor de referência. O servidor de autorização proposto nesse modelo assume uma função semelhante ao monitor de referência do lado servidor (SRM) apresentado em [PARK 2002], a diferença é que o servidor de autorização proposto atua indiretamente no processo no controle de uso sobre conteúdos digitais. O servidor de autorização não determina como será o acesso a conteúdos pertencentes a seu domínio, mas para todos os conteúdos que estarão sob domínio dos usuários. Esse controle distribuído ocorre porque a entidade de autorização na modelo proposto funciona como servidor de distribuição de direitos de acesso.

O processo de autorização previsto nesse trabalho é executado por uma aplicação de *middleware*. Essa aplicação identifica requisições que chegam no servidor de autorização e executa o processo de decisão através da avaliação das políticas de controle de acesso. A aplicação é também responsável por distribuir os direitos que um usuário possui sobre um determinado conteúdo.

5.5. CONCLUSÃO DO CAPITULO

Este capítulo apresentou uma proposta de um modelo de autorização e distribuição de direitos de acesso para conteúdos digitais. Também demonstrou o processo de avaliação de políticas de controle de acesso para execução do processo de autorização. O capítulo também apresentou no modelo da arquitetura, a distribuição de direitos para suporte a um serviço de distribuição de licenças.

O uso de padrões de segurança XML abertos pode realmente facilitar a execução de controle de uso de conteúdos digitais em um contexto DRM, na medida que estabelecem a troca de mensagens seguras entre as entidades do sistema. O modelo de autorização e distribuição de direitos de acesso serve como suporte para aplicações responsáveis por controlar uso de conteúdos digitais, livrando essas aplicações da execução de processos de decisão.

Finalmente, o uso de uma linguagem padrão para definição de políticas de controle de acesso em XML determina interoperabilidade em ambientes de controle de acesso distribuído.

6. IMPLEMENTAÇÃO

No capítulo anterior foi apresentado um modelo para distribuição de direitos de acesso através da avaliação de políticas de segurança. Este capítulo descreve a implementação do protótipo e os resultados obtidos através dos testes. Também descreve como foi definido e implementado um *framework*[3] que permite a execução de todos os processos necessários, para a distribuição de direitos de acesso, propostos na estrutura do modelo definido.

Para testar o protótipo foi escolhido um ambiente para distribuição de direitos sobre *E-Books*. A implementação do protótipo recebe como entrada uma requisição de decisão de autorização SAML, avalia a requisição através de políticas de segurança implementadas em XACML e produz como saída uma resposta de decisão de autorização SAML.

6.1. ESTRUTURA DE UMA APLICAÇÃO DE AUTORIZAÇÃO E DISTRIBUIÇÃO DE DIREITOS (AADD)

A aplicação AADD é dividida em diferentes módulos que executam as funções referentes ao PDP, PRP e PIP e outros módulos responsáveis por receber, enviar e formatar mensagens. A figura 36 demonstra a estrutura da aplicação de autorização e distribuição de direitos.

O módulo *Listen/Response* é responsável pelo recebimento de requisições de autorização sobre direitos (passo 1) e por enviar as mensagens de resposta sobre a decisão de autorização (passo 10). O módulo de *Encapsulamento/Desencapsulamento* XML é responsável por coletar as informações contidas nas requisições SAML que chegam e repassá-las ao controle de decisão no formato desejado (passo 3). Ele também

---

[3] *Do ponto-de-vista do desenvolvimento, um framework é definido como uma arquitetura de software reutilizável em termos de contratos de colaboração entre classes abstratas e um conjunto de pontos adaptáveis, ou hot spots [CODENIE 1997].*

é responsável por formatar a decisão de autorização em um padrão SAML de mensagem de resposta desejada (passo 8).

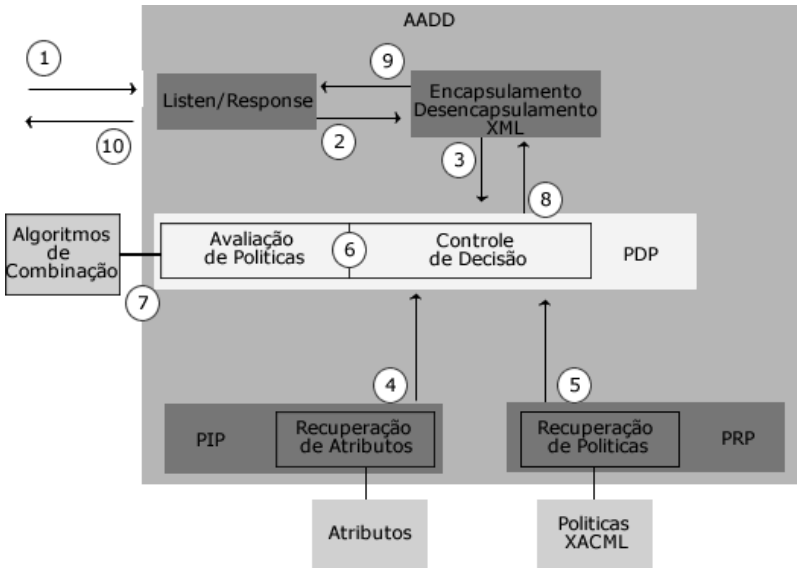

Figura 36 – Estrutura da AADD.

O módulo de recuperação de políticas agrupa as políticas necessárias requeridas pelo controle de decisão (passo 5). O módulo de recuperação de atributos disponibiliza valores de atributos, que não estão presentes na requisição de autorização, necessários para recuperação de políticas envolvidas no processo de decisão (passo 4). Finalmente o controle de decisão efetua a avaliação de políticas (passo 6) através do uso de algoritmos de combinação (passo 7) que geram a decisão de autorização sobre os direitos.

6.2. FERRAMENTAS UTILIZADAS PARA O DESENVOLVIMENTO

O ambiente de desenvolvimento escolhido para a implementação do protótipo foi Java [JAVA]. A linguagem Java foi escolhida por representar uma ferramenta que permite a sua portabilidade entre diferentes plataformas e também pela existência de pacotes de desenvolvimento e APIs para uso de tecnologias como o SAML, o XACML e *Parsers* XML.

Os pacotes de desenvolvimento utilizados foram o OpenSAML [OPEN SAML] versão1.0, SunXACML [SUN XACML] e o parser Xerces. O pacote OpenSAML, foi desenvolvido pela universidade do estado de Ohio nos Estados Unidos (*The Ohio State University*) [OPEN SAML], e apresenta um conjunto de classes para representar as diferentes operações do padrão SAML 1.0, permitindo a criação de requisições e

respostas SAML de maneira fácil e rápida. O pacote SunXACML, foi projetado e desenvolvido pela empresa *Sun Microsystems*, e apresenta classes que implementam os processos de criação de requisições e respostas XACML bem como a avaliação de políticas. O pacote também apresenta classes para geração de políticas XACML. O *parser* XML Xerces é requerido para esses dois pacotes de desenvolvimento, porque implementa a tecnologia de padrão para interfaces XML baseadas em objetos, DOM3, que monta a árvore de elementos de um documento XML.

O protótipo foi desenvolvido como um *framework* para aplicações Java, e contém as diversas classes que representam os diferentes módulos da estrutura de uma aplicação AADD (figura 36). A implementação dessas classes permite que outras aplicações possam utilizá-las de maneira individual. Assim, desenvolvedores poderiam utilizar as classes de maneira a implementar a melhor solução para suas necessidades.

A figura 37 apresenta o *framework* e a relação entre as ferramentas utilizadas e as classes implementadas para a confecção do protótipo.

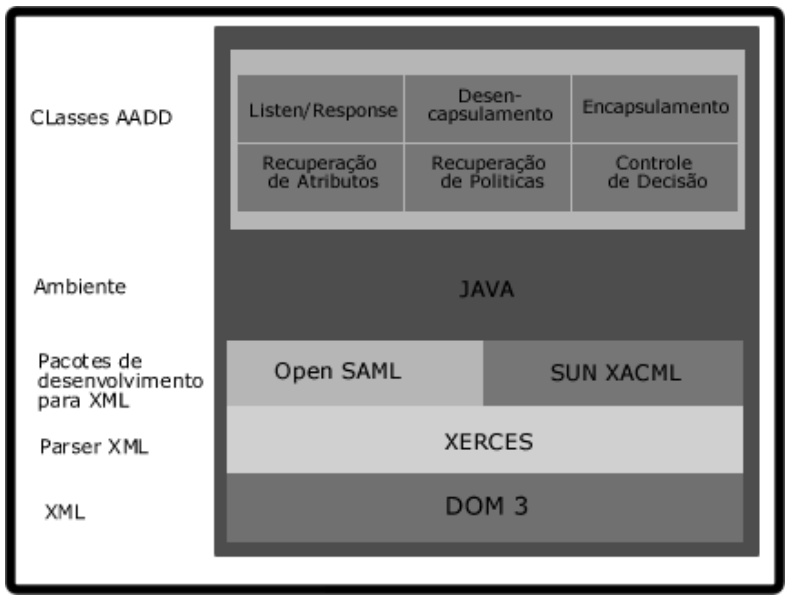

Figura 37 – *Framework* para confecção do protótipo.

6.3. CLASSES E MÉTODOS IMPLEMENTADOS

As classes que compõe o *framework* desenvolvido são as seguintes: *Desencapsulamento, Encapsulamento, ControleDecisao, RecAtributos, RecPoliticas* e *ListenResponse.*

A classe *Desencapsulamento* tem como função principal carregar e validar requisições SAML. Esta classe possui dois métodos chamados de *AbreEnvelopeSoap* e *AbreRequestSAML*.

A classe *Encapsulamento* tem como função principal transformar uma resposta de decisão XACML em uma resposta de decisão de autorização SAML. Essa classe possui oito métodos, destacando-se como principal método o *GeraResponseSAML*, que é responsável por gerar a resposta de decisão de autorização SAML. Os outros métodos são complementares ao método *GeraResponseSAML*.

A classe *ControleDecisao* tem como funções principais transformar uma requisição SAML em uma requisição XACML e avaliar políticas mediante a requisição criada. Nesta classe são definidos sete métodos, os principais métodos são: o método *AvaliaPolitica*, que faz a avaliação de políticas; o método *SAMLToXACML*, que reúne as informações necessárias para geração da requisição XACML através do método *GeraRequestXACML*, que efetivamente cria a requisição. Os outros métodos são complementares a esses três métodos principais.

A classe *RecAtributos* tem como função principal a recuperação de atributos através da criação de requisições de atributos SAML e a interpretação de respostas de atributos SAML. Esta classe possui cinco métodos onde o principal método é o *GetSubjectAttribute*, responsável por recuperar atributos de sujeitos presentes em uma resposta de atributos SAML.

A classe *RecPoliticas* tem como função principal a recuperação do conjunto de políticas que serão avaliadas para o processo de autorização. Esta classe possui um único método, *RecuperaPoliticas*, que carrega um conjunto de arquivos de políticas XACML.

A classe *ListenResponse* tem como função receber e transferir mensagens SOAP. Esta classe realiza a comunicação com o ambiente externo ao domínio o qual pertence.

6.4. IMPLEMENTAÇÃO DE POLÍTICAS DE SEGURANÇA

As políticas de segurança utilizadas para os testes no ambiente de distribuição de direitos foram implementadas utilizando as especificações do padrão XACML 1.0, e validadas pela ferramenta de validação de políticas contida no pacote de desenvolvimento Java SunXACML.

A política utilizada no processo de validação foi implementada em um único arquivo XML. A política foi definida, em seu objeto *target*, como sendo uma política que se aplica a todos os usuários do sistema e a qualquer ação. O recurso ao qual a política se aplica é específico e foi definido como uma URI nomeada como "http://www.livropdf.com/".

Nessa política, de acordo com as regras estabelecidas, os usuários e ações são definidos de acordo com a classe a qual pertencem. Assim usuários definidos como classe "A" possuem o direito de impressão. Os usuários definidos como pertencentes à classe "B" possuem apenas o direito de visualização. A código completo da implementação dessa política é apresentado no Anexo 1 deste trabalho.

O algoritmo de avaliação e combinação, de regras de controle de acesso, utilizado para a política de segurança foi o *permit overrides rule combining algorithm,* definido no padrão XACML 1.0. Esse algoritmo avalia todas as regras pertencentes a uma política, e retorna como resultado final o valor *Permit* se o resultado de avaliação de qualquer uma das regras retornar um valor *Permit*. Assim se a avaliação de nenhuma regra retornar um *Permit* o resultado final é *Deny*.

6.5. FUNCIONAMENTO DO FRAMEWORK

6.5.1. Diagrama de Seqüência para o framework

O funcionamento do *framework*, que é utilizado para o desenvolvimento de uma aplicação AADD, pode ser visualizado através do diagrama de seqüência apresentado na figura 38. Conforme o diagrama de seqüência, a aplicação AADD possui como entrada uma mensagem SOAP, contendo uma requisição de decisão de autorização SAML. A partir dessa entrada a aplicação invoca um método da classe *Desencapsulamento* para carregar a requisição SAML e validá-la. Depois, a aplicação através da classe *Controle de Decisão*, invoca métodos responsáveis por transformar a requisição SAML em uma requisição de decisão XACML. É importante observar que o processo de transformação das requisições se torna necessário para realizar o processo de avaliação das políticas de segurança. É nessa etapa também que ocorre o processo de recuperação de atributos que não estão presentes na requisição SAML de entrada, essa recuperação de atributos é feita através da classe *Recuperação de Atributos* que pode emitir uma requisição de atributos SAML, e receber como resposta também um arquivo

XML contendo uma asserção de atributos SAML. A aplicação, então de posse dos atributos, gera uma requisição de decisão XACML e pode avaliar corretamente as políticas.

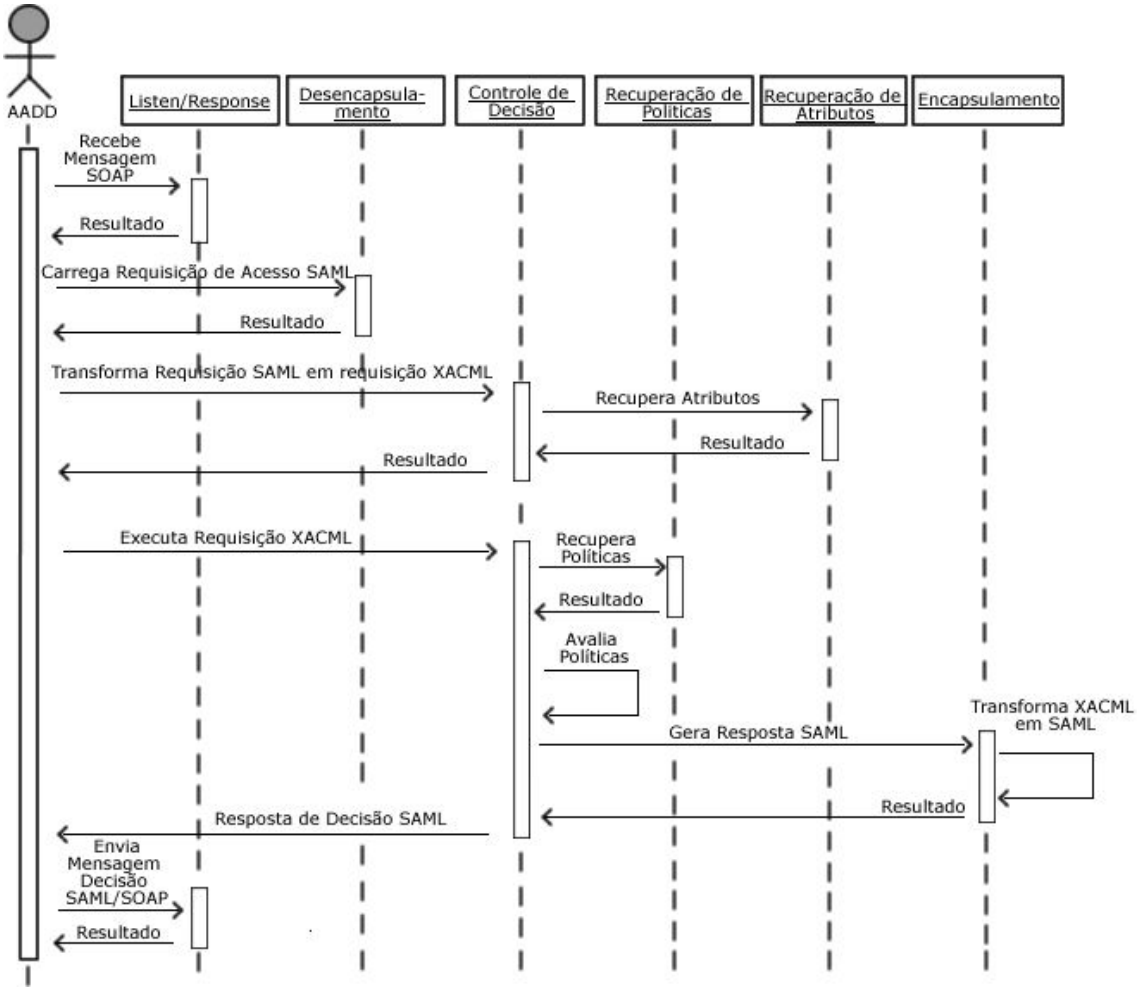

Figura 38 – Diagrama de Seqüência para o *framewok*.

De posse da requisição XACML, a aplicação invoca outro método da classe *Controle de Decisão,* encarregado de avaliar a requisição e retornar a resposta de decisão XACML. A classe *Controle de Decisão* pode ser configurada de modo a determinar quais são os atributos necessários, além da identificação do sujeito, para efetuar a recuperação e avaliação das políticas corretamente. Esta configuração pode ser feita diretamente na classe ou pela aplicação que está utilizando a classe, em tempo real, através de atributos definidos para este fim.

Depois, a aplicação invoca um método da classe *Encapsulamento* que é responsável por transformar a resposta de decisão XACML em uma resposta SAML, esta resposta deve conter uma asserção de decisão de autorização SAML. Finalmente , a

resposta SAML é enviada pela aplicação, até seu requisitante, através da classe *Listen/Response*.

6.6. PROTÓTIPO DESENVOLVIDO

O protótipo é uma implementação de um *framework* que visa facilitar o desenvolvimento de um ambiente para uso de um monitor de referência do lado do servidor, conforme o modelo proposto. Visando observar o funcionamento do *framework*, adotou-se a idéia de simular um sistema real de distribuição de direitos sobre *E-Books*. Esse sistema é responsável por distribuir os direitos de acesso para todos os conteúdos pertencentes a um domínio genérico, no caso "http://www.livropdf.com/". Assim, foi determinado que nesse sistema a existência de dois tipos de usuários, pertencentes às classes "A" e "B". Os direitos de acesso a qualquer conteúdo pertencente ao domínio estabelecido são determinados através da política de segurança implementada (Anexo 1).

Com o objetivo de simular o funcionamento de uma aplicação AADD, através do *framework* desenvolvido, foi implementada uma interface gráfica amigável, em linguagem Java. Com essa interface foi possível determinar os dados de entrada, como o arquivo que possui a requisição SAML e o arquivo de políticas, e também o nome do arquivo onde será gerada a resposta SAML.

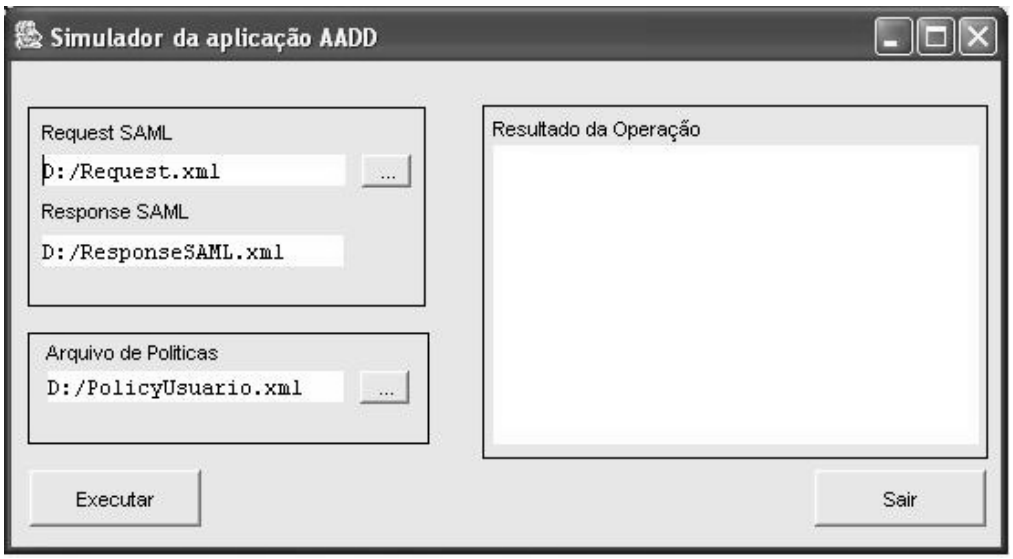

Figura 39 – Interface utilizada para validação do protótipo.

A figura 39 apresenta a interface criada com o objetivo de simular a entrada de dados e observar o funcionamento das classes. Nessa interface o campo "Request

SAML" indica o local (*path*) onde se encontra um arquivo contendo uma requisição SAML de entrada. O campo "Response SAML" contém o local e nome onde deve ser gravada a resposta de decisão SAML. O campo denominado como "Arquivo de Políticas" determina o local e o nome do arquivo de políticas a ser avaliado. A interface ainda apresenta um espaço reservado para a demonstrar o resultado final do processo.

A aplicação AADD inicialmente instancia a classe *Controle de decisão* e invoca o método *SAMLToXACML*. Este método converte uma requisição de decisão de autorização SAML entrada para uma requisição de decisão XACML. A figura 40 apresenta a requisição SAML de entrada utilizada na simulação. A requisição indica que um usuário identificado como "Valerio" está requisitando direito de visualização de algum conteúdo pertencente ao domínio "http://www.livropdf.com/".

```
- <Request xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" IssueInstant="2003-12-
    12T12:34:53Z" MajorVersion="1" MinorVersion="1"
    RequestID="cb87afb100b1e85aee6dd2fa38583eb4">
  - <AuthorizationDecisionQuery Resource="http://www.livropdf.com/">
    - <Subject xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
        <NameIdentifier>Valerio</NameIdentifier>
      </Subject>
      <Action xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
        Namespace="urn:oasis:names:tc:SAML:1.0:assertion">View</Action
        >
    </AuthorizationDecisionQuery> </Request>
```

Figura 40 - *Request* SAML de entrada.

Conforme visto anteriormente, para efetuar a conversão de uma requisição de autorização SAML em uma requisição XACML, é necessária a coleta de alguns atributos necessários para a execução de avaliação de uma política. Então o método *SAMLToXACML* internamente invoca o método *GetSubjectAttribute* que pertence a classe *Recuperação de Atributos*. O método *GetSubjectAttribute* extrai de uma resposta de atributos de sujeitos SAML, os valores de atributos necessários para a geração da requisição XACML. A figura 41 apresenta a resposta de atributos de sujeitos SAML, que apresenta em seu conteúdo a informação sobre a classe a qual o usuário, identificado como "Valerio", pertence.

```
- <Response xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
    InResponseTo="fd364e1eb83fd9376ef26749012f0852" IssueInstant="2004-01-
    12T18:56:43Z" MajorVersion="1" MinorVersion="1"
    Recipient="www.emissor.com.br"
    ResponseID="b60a8ddd11db741c9d399dada9a290e4">
  - <Status>
      <StatusCode Value="samlp:Success" />
    </Status>
  - <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
      AssertionID="f1d47e98ba79869c1db22e934213e7e0" IssueInstant="2004-
      01-12T18:56:43Z" Issuer="www.emissor.com.br" MajorVersion="1"
      MinorVersion="1">
      <Conditions NotBefore="2004-01-12T18:56:43Z" NotOnOrAfter="2004-01-
        12T18:56:43Z" />
    - <AttributeStatement>
      - <Subject>
          <NameIdentifier>Valerio</NameIdentifier>   </Subject>
      - <Attribute AttributeName="Classe-ID"
          AttributeNamespace="www.teste.com.br">
          <AttributeValue>A</AttributeValue>
        </Attribute>
      </AttributeStatement>
    </Assertion>
  </Response>
```

Figura 41 – Resposta de atributos SAML

De posse desse atributo adicional, o método *SAMLToXACML* então finaliza a o processo de geração da requisição XACML invocando o método *GeraRequestXACML*, que gera a requisição XACML. A requisição XACML gerada como resultado da execução do método *SAMLToXACML* é apresentada na figura 42.

A aplicação, de posse da requisição XACML, invoca o método *AvaliaPolitica*. A aplicação passa como parâmetros a requisição XACML (figura 42) e o arquivo de políticas (Anexo1). O método *AvaliaPolitica* seleciona as regras referentes aos usuários específicos da classe “A”, e depois efetua o processo de combinação das regras de autorização presentes na política obedecendo a um algoritmo de combinação de regras definido no cabeçalho da política XACML. De acordo com a política implementada o resultado dessa combinação de regras tem como resultado final um efeito “*Permit*”. Assim, é gerada uma resposta de decisão XACML.

```
- <Request>
  - <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-
        category:access-subject">
    - <Attribute AttributeId="Classe-Id"
          DataType="http://www.w3.org/2001/XMLSchema#string"
          Issuer="www.valerio.com">
        <AttributeValue>A</AttributeValue>
      </Attribute>
    - <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-
          id" DataType="http://www.w3.org/2001/XMLSchema#string"
          Issuer="www.valerio.com">
        <AttributeValue>Valerio</AttributeValue>
      </Attribute>
  </Subject>
  - <Resource>
    - <Attribute
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="http://www.w3.org/2001/XMLSchema#anyURI">
        <AttributeValue>http://www.livropdf.com/</AttributeValue>
      </Attribute>
  </Resource>
  - <Action>
    - <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
          DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue>Print</AttributeValue>
      </Attribute>
    </Action> </Request>
```

Figura 42 – Requisição XACML

Após o processo de avaliação da política a aplicação AADD instancia a classe *Encapsulamento* e invoca o método *GeraResponseSAML*. Esse método transforma a resposta de decisão XACML em uma resposta SAML.

Efetivamente, a aplicação, produz como resultado final um arquivo XML contendo um *Response* SAML, formado por uma asserção de decisão de autorização SAML. A figura 43 apresenta a resposta produzida pela aplicação, onde a *tag AuthorizationDecisionStatement* apresenta a decisão de autorização e o recurso para o qual a decisão será aplicada. A resposta SAML também identifica o sujeito que possui o direito definido pela decisão.

```
- <Response xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
    InResponseTo="cb87afb100b1e85aee6dd2fa38583eb4"
  IssueInstant="2004-01-19T18:38:03Z" MajorVersion="1" MinorVersion="1"
    Recipient="www.valerio.com"
    ResponseID="ff184470e338a2aa27c0418e13ae8e4e">
- <Status>
  <StatusCode Value="samlp:Success" />
  </Status>
-<Assertion  xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
    AssertionID="efcc85388efbc877c8ecead7505c4a60"
    IssueInstant="2004-01-19T18:38:03Z"  Issuer="www.valerio.com"
    MajorVersion="1" MinorVersion="1">
  <Conditions NotBefore="2004-01-19T18:38:03Z"
    NotOnOrAfter="2004-01-19T18:38:03Z" />
    -<AuthorizationDecisionStatement
        Decision="Permit" Resource="http://www.livropdf.com/">
-  <Subject>
   <NameIdentifier>Valerio</NameIdentifier>
   </Subject>
 <Action Namespace="urn:oasis:names:tc:SAML:1.0:assertion">View</Action>
</AuthorizationDecisionStatement>
</Assertion>
</Response>
```

Figura 43 – *Response* SAML de saída.

6.7. CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou aspectos relevantes ao desenvolvimento de aplicações que visam a distribuição de direitos de acesso. Apresentou as ferramentas e estrutura utilizada para o desenvolvimento do protótipo, cujo principal objetivo era testar a aplicabilidade do modelo de distribuição de direitos proposto nesta dissertação.

O *framework* desenvolvido promove a interoperabilidade entre domínios *web* e a aplicação da estrutura de avaliação de políticas e distribuição de direitos em diferentes ambientes.

7. CONCLUSÕES E TRABALHOS FUTUROS

Este dissertação apresentou uma proposta de um modelo de autorização e distribuição de direitos de acesso para conteúdos digitais, utilizando a estrutura oferecida pela *web*, através do uso de padrões de segurança XML e *Web Services*, para o suporte a distribuição de licenças em sistemas DRM. O processo de avaliação de políticas de controle de acesso para execução do processo de autorização, também foi apresentado. O trabalho também apresentou no modelo como ocorre a distribuição de direitos de acesso.

7.1. REVISÃO DAS JUSTIFICATIVAS

O uso de padrões de segurança XML abertos pode realmente facilitar a execução de controle de uso de conteúdos digitais em um contexto DRM, na medida que estabelecem a troca de mensagens seguras entre as entidades do sistema. Isso supre a necessidade de controlar o uso de conteúdos digitais, para evitar fraudes e violação de direitos autorais bem como a necessidade de garantir a segurança na troca de informações que disponibilizam direitos.

O modelo de autorização e distribuição de direitos serve como suporte para aplicações responsáveis por controlar uso de conteúdos digitais, livrando essas aplicações da execução de processos de decisão.

O uso de uma linguagem padrão para definição de políticas de controle de acesso em XML possibilita a interoperabilidade em ambientes de controle de acesso distribuído, e supre a necessidade de definir uma base de autorização que determine que tipos de direitos sujeitos possuem sobre objetos.

7.2. VISÃO GERAL DO TRABALHO

Este trabalho abordou os conceitos básicos de segurança computacional ressaltando aspectos relacionados ao controle de acesso a ambientes e recursos distribuídos. Também apresentou noções de W*eb Services* e os principais padrões de

segurança desenvolvidos para suprir as necessidades de segurança impostas por essa arquitetura de serviços, através do padrão XML.

O trabalho descreveu o uso de padrões de segurança XML em uma estrutura de distribuição de direitos de acesso como suporte a uma estrutura de Controle de Uso de conteúdos digitais através da distribuição de licenças. O *framework* desenvolvido proporciona a sua utilização em outras aplicações.

Finalmente o trabalho apresentou a implementação do protótipo, que proporciona a visualização de como ocorre o processo de distribuição de direitos bem como o processo de avaliação de políticas.

7.3. CONSIDERAÇÕES SOBRE OS TRABALHOS RELACIONADOS

Reiner Kraft [KRAFT 2002] em seu trabalho aborda conceitos para definição de Processadores Controle de Acesso (ACPs) distribuídos, controlando diretamente o acesso a recursos. Em sua proposta, Kraft define que cada ACP é encarregado de controlar o acesso de um determinado conjunto de recursos, e o controle de acesso é baseado em listas de controle de acesso (ACL) presentes em cada ACP. Da mesma maneira Park e Sandhu [PARK 2002] em seu trabalho abordam um monitor de referência (SRM) fazendo a mediação de recursos pertencentes a seus domínios, executando o controle de acesso no lado servidor.

Diferente disso, o modelo proposto nesta dissertação não faz um controle de acesso direto sobre os recursos, apenas determina e distribui os direitos de acesso. Estes serão aplicados por outras estruturas de controle de acesso. A modelo define o uso de políticas de segurança para estabelecer direitos de acesso ao conteúdo protegido, através do uso do de uma linguagem padrão de definição de política, o XACML. A definição de políticas também permite o uso de mecanismos de controle de acesso tradicionais.

Reiner Kraft [KRAFT 2002] também não define em seu trabalho como ocorrem as trocas de mensagens entre diferentes ACPs. As trocas de mensagens de autorização são feitas utilizando o apenas o protocolo SOAP, não incorrendo em um formato específico. Já o modelo apresentado nesta dissertação define como ocorre o processo de trocas de mensagens através do uso do padrão SAML. O uso do SAML permite a interoperabilidade completa entre domínios *web* que necessitam implementar controle de acesso distribuído.

As propostas apresentadas em [HADA 2000b], [DAMIANI 2002] e [GABILON 2001], buscam definir uma linguagem de especificação de políticas de controle de acesso baseadas em XML para o controle de acesso de grão-fino a recursos. Esse tipo de controle de acesso é facilmente aplicado ao modelo apresentado nessa dissertação, porque o padrão XACML permite a definição de políticas que determinem acesso a partes de recursos, desde que esses recursos apresentem uma estrutura conhecida para a aplicação que efetua o controle de acesso do lado cliente, no caso um controle de uso.

7.4. PRINCIPAIS CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

a) Definição de um modelo de distribuição de direitos de acesso aplicado a DRM;
b) Definição do processo de autorização através do uso de políticas de segurança definidas com XML;
c) Integração entre entidades de sistemas DRM utilizando padrões de segurança XML;
d) Interoperabilidade entre domínios *web* através do uso padrões de trocas de mensagens XML;
e) Distribuição de direitos através de mensagens XML;
f) Definição de políticas que implementam mecanismos de controle de acesso tradicional;
g) Definição de um modelo para avaliação de políticas de segurança baseadas em XML;
h) Definição de um *framework* de apoio aplicações distribuídas que visam o controle de uso;
i) Integração dos padrões XACML e SAML para a distribuição direitos de acesso.

7.5. PERSPECTIVAS FUTURAS

A definição de uma arquitetura completa para a distribuição de direitos depende de várias ferramentas e abordagens que definam a melhor maneira de executar o processo de suporte ao controle de uso através de distribuição de licenças.

Mensagens, que carregam direitos de acesso, podem ter sua integridade e autenticidade comprometidas. Assim, como trabalho futuro pode ser definido a inclusão de métodos para implementar a integridade e autenticidade das requisições e respostas SAML, utilizando os padrões como o XML-*Signature* e XML *Encryption*.

A avaliação do desempenho do protótipo desenvolvido, em diferentes ambientes e condições, pode ser vista como uma maneira de melhorar o modelo proposto, quando o desempenho for fator crítico para a aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

[AMOROSO 1994] AMOROSO Edward. **Fundamentals of Computer Security Tecnology** . Ed. Pretice Hall, Inc Englewood Ciffs – New Jerssey, 1994.

[APPARAO 1998] APPARAO V., BYRNE S., CHAMPION M. et al. **Document Object Model (DOM) Level 1 Specification**. W3C Recommendation, October 1998. Disponível em http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/ .

[BEECH 2001] BEECH David; MALONEY Murray; MENDELSOHN Noah et al. **XML Schema Part 1: Structures**. W3C Recommendation, May 2001. Disponível em http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ .

[BELL E LAPADULA 1976] BELL D. Elliot, LAPADULA Leonard J. **Security Computer Systems: Unifield exposition and Multics Interpretation**. MITRE Techinical Report MTR-2297 Ver. 1, Mitre Corporation, BedFord, MA, March 1976.

[BELLWOOD 2002] BELLWOOD Tom; CLÉMENT Luc; RIEGEN Claus von. **UDDI Version 3.0**. UDDI Spec Technical Committee Specification, 19 July 2002. Disponível em http://uddi.org/pubs/uddi_v3.htm .

[BIBA 1977] BIBA Kenneth J. **Integrity Considerations for Secure Computer System**. MITRE Technical Report MTR –3153, MITRE Corporation, Besford, MA, April 1977.

[BOYER 2001] J. Boyer. **Canonical XML**. W3C Recommendation, March 2001. Disponível em http://www.w3.org/TR/2001/REC-xml-c14n-20010315 .

[BOX 2000] BOX Don; EHNEBUSKE David; KAKIVAYA Gopal et al .**Simple Object Access Protocol (SOAP) 1.1**, W3C Note 08 May 2000. Disponível em http://www.w3.org/TR/SOAP/ .

[BRAY 2000] BRAY Tim; PAOLI Jean; MALER Eve et al. **Extensible Markup Language (XML) 1.0 (Second Edition)**. World Wide Web Consortium (W3C) Recommendation, October 2000. Disponível em http://www.w3c.org/TR/REC-xml .

[BRAY 1999] BRAY Tim; HOLLANDER Dave; LAYMAN Andrew. **Namespaces in XML**. W3C Recommendation, January 1999. Disponível em

http://www.w3.org/TR/1999/REC-xml-names-19990114/ .

[BREWER e NASH 1989] BREWER D.F.C., NASH M.J. **The Chinese Wall security policy**. In Procedings of the 1989 IEEE symposium on Security and Privacy, pp 206-214, 1989.

[BRICKLEY 2000] BRICKLEY Dan; GUHA R.V. **Resource Description Framework (RDF) Schema Specification 1.0**. W3C Candidate Recommendation. March 2000. Disponível em http://www.w3.org/TR/2000/CR-rdf-schema-20000327/ .

[CHONG 2002] CHONG C. N.; VANBUUREN R.; HARTEL P. H. et al. **Security attribute based digital rights management.** In Joint Int. Workshop on Interactive Distributed Multimedia Systems/Protocols for Multimedia Systems (IDMS/PROMS),p.toappear, Springer-Verlag, Berlin, November 2002.

[CHRISTENSEN 2001] CHRISTENSEN Erik; CURBERA Francisco, MEREDITH Greg et al. **Web Services Description Language (WSDL) 1.1**. W3C Note 15 March 2001. Disponível em http://www.w3.org/TR/wsdl .

[CLARK 1999a] CLARK James; DEROSE Steve. **XML Path Language (XPath) Version 1.0**. World Wide Web Consortium (W3C), November 1999. Disponível em http://www.w3c.org/TR/xpath .

[CLARK 1999b] CLARK James. **XSL Transformations (XSLT) Version 1.0**. World Wide Web Consortium (W3C), November 1999. Disponível em http://www.w3c.org/TR/xslt .

[CLARK e WILSON 1987] CLARK David; WILSON David. **A Comparison of Comercial and Military Computer Security Policies**. In Procedings of the IEEE Symposium on Security and Privacy, p. 184-194, Oakland , CA, May 1987.

[CODENIE 1997] CODENIE W.; HONDT K.; STEYAERT P. et al. **Custom Applications to Domain-Specific Frameworks**. Communications of the ACM, 40(10), 71-77, 1997.

[DAMIANI 2000] DAMIANI Ernesto; VIMERCATI Sabrina; SAMARATI Pirangela et al. **Securing XML Documents**, In Proc. of the 2000 International Conference on Extending Database Technology (EDBT2000), Konstanz, Germany, March 27-31, 2000.

[DAMIANI 2002] DAMIANI Ernesto; VIMERCATI Sabrina; SAMARATI Pirangela et al. **A fine-grained access control system for XML documents**. ACM

Transactions on Information and System Security (TISSEC) May 2002, pp. 169-202, Volume 5, Issue 2.

[DEITEL 2002] DEITEL H.M , DEITEL P.J, GADZIC J. P. et al. **Java Web Services For Experienced Programers**. Printed by Prentice Hall, Uper Saddle River , New Jersei 07458, 2003.

[DOD 1985] US Department Of Defense. **DoD Trusted Computing System Evolution Criteria (The Orange Book)**. DOD 5200.28-STD, 1985.

[EASTLAKE 2002a] EASTLAKE Donald; REAGLE Joseph; SOLO David. **XML-Signature Syntax and Processing**. W3C Recommendation, February 2002. Disponível em http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ .

[EASTLAKE 2002b] EASTLAKE Donald; REAGLE Joseph. **XML Encryption Syntax and Processing**. W3C Candidate Recommendation, August 2002. Disponível em http://www.w3.org/TR/2002/CR-xmlenc-core-20020802/ .

[EBXML 2001] ebXML Web Site, 2001.Disponivel em http://www.ebXML.org.

[EDDON 1998] EDDON G. et al. **Inside Distributed COM**, Microsoft Press, 1998.

[FILIPPIN 2004] FILIPPIN Cleber V. **Uma Proposta de Arquitetura para Distribuição e Gerenciamento de Licenças de Direitos Digitais**, Dissertação de Mestrado INE\UFSC, Brasil, Fevereiro, 2004.

[FRANKSTON 1998] FRANKSTON Charles; BRAY Tim; MALHOTRA Ashok. **Document Content Description for XML**. Submission to the World Wide Web Consortium, 31-July-1998.

[GABILON 2001] GABILLON Alban; BRUNO Emmanuel. **Regulating Access to XML documents.** Fifteenth Annual IFIP WG 11.3 Working Conference on Database Security. Niagara on the Lake, Ontario, Canada July 15-18, 2001.

[GODIK 2003] GODIK Simon., MOSES Tim. **eXtensible Access Control Markup Language (XACML) Version 1.0.** OASIS Standard, 18 February 2003.

[GOLLMANN 1999] GOLLMANN Dieter. **Computer Security**. John Wiley and Sons, 1999.

[HADA 2000a] HADA Satoshi; KUDO Michiharu. **XML Access Control Language: Provisional Authorization for XML Documents**. October 16, 2000. Disponível em http://www.trl.ibm.com/projects/xml/xacl/xacl-spec.html

[HADA 2000b] HADA Satoshi; KUDO Michiharu. **XML Document Security based on Provisional Authorisation**. In Proceedings of the 7th ACM Conference on Computer and Communications Security. November, 2000, Athens Greece, p. 87-96.

[HALLAM-BAKER 2003] HALLAM-BAKER Phillip. **XML Key Management Specification (XKMS) Version 2.0**. W3C Working Draft, April 2003. Disponível em http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ .

[ISO 1988] International Organization for Standarization. **ISO 7498-2 Basic Reference Model for Open Systems Interconnection (OSI) Part 2: Security Architeture**. Geneva, Switzerland , 1988.

[ISO/IEC 15408-1, 1999] ISO/IEC 15408-1:1999(E) – Part 1: Introduction and general model. *In: Information Technology – Security Techniques – Evatuation Criteria for IT Security*, First edition, ISO/IEC, December 1999.

[JAVA] Sun Microsystems, Inc. **Java Technology**. Disponível em http://java.sun.com/

[JOHNER 1998] JOHNER Heiz; BROWN Larry; HINNER Franz-Stefan et al. ***Understanding LDAP*** .IBM SG24-4986-00, June 1998. Disponível em http://www.reedbooks.ibm.com

[JURIC 2000] JURIC Matjaz B.; ROZMAN Ivan, NASH Simon . **Java 2 Distributed Object Middleware Performance Analysis and Optimization**, ACM SIGPLAN Notices, No. 8, August 2000.

[KRAFT 2002] KRAFT Reiner. **Designing a distributed access control processor for network services on the Web**. Proceedings of the *ACM workshop on XML security 2002*, Fairfax, VA November 22 - 22, 2002.

[LAI 1992] LAI X. **On The Design and Security of Block Ciphers**. ETH series in Information Processing, v.1, Konstans: Hartung-Gorre Verlag, 1992.

[LANDWEHR 1983] LANDWEHR C. E. **The best avaliable technologies for computer security**. ACM Computing Surveys ACM, v.16, n. 7 , p 86-95, July 1983.

[LAMPSON 1971] LAMPSON Butler W. **Protection**. In: *5th Princeton Symposium on Information Sciences and Systems*, March 1971. Reprinted in *ACM Operating Systems Review*, v.8, n.1, p. 18-24, 1974

[LAYMAN 1998] LAYMAN Andrew; JUNGE Edward; MALER Eve. **XML-Data**. W3C Note, Jan 1998. Disponível em http://www.w3.org/tr/1998/Note-XML-Data

[MALER 2002] MALER Eve; HALLAM-BAKER Phillip. **Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.0**, OASIS Standard, 5 November 2002.

[MALER 2003] MALER Eve; MISHRA Prateek.; PHILPOTT Rob. **Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1**, OASIS Standard, 18 July 2003.

[MEGGINSON 2000] MEGGINSON David. **SAX 2.0: The Simple API for XML**. 2000. Disponível em http://www.megginson.com/SAX.

[MURATA 2001] MURATA Makoto; CLARK James. **RELAX NG Specification**. OASIS, 2001.

[OMG 1999] Object Management Group. **CORBA 2.3.2 Specification**. *OMG Document Number 99-10-07*, October 1999.

[OPEN SAML] The Ohio State University Corporation for Advanced Internet Development. **The OpenSAML version 1**. 2002. Disponível em http://wayf.internet2.edu/opensaml/java/doc/api/index.html

[PARK 2002] PARK Jaehong, SANDHU Ravi. **Towards usage control models: beyond traditional access control**. Proceedings of the *seventh ACM symposium on Access control models and technologies 2002, Monterey*, California, USA June 03 - 04, 2002.

[SANDHU e SAMARATI 1994] SANDHU Ravi S., SAMARATI Pirangela **Acess Control : Principles and Practice**. IEEE Communications, v.32, n.9, p.40-48 , 1994.

[SANDHU 1993] SANDHU Ravi S. **Lattice-Based Access Control Models** . IEEE Computer , v.26 , n.11, p.9-19, November 1993.

[SANDHU 1998] SANDHU Ravi S. **Role-Based Access Control**. In Selkowitz editor, *Advances in Computers*, v.46 , Academic Press , 1998.

[SCHNEIER 1993] SCHNEIER B. **Description of a New Variable-Length Key, 64-bit Block Cipher (BlowFish)**. *Proceedings, Workshop on fast Software Encryption*, December 1993; published by Springer-Verlag.

[SIMEON 2002] SIMEON Jerome; WADLER Philip. **The Essence of XML**. FLOPS 2002, Aizu, Japan, 15-17 September 2002

[SMID e BRANSTAD 1988] SMID M. E.; BRANSTAD D. K.. **The Data Encryption Standard: Past and Future**, *In: Proceedings of The IEEE* , v.76, n.5, p.550-559, May 1998.

[SOARES 1999] SOARES Luiz Fernando; LEMOS Guido; COLCHER Sérgio. **Redes de computadores: Das LANs, MANs e WANs às redes ATM**. Editora Campus, 1995.

[STALLINGS 1999] STALLINGS, W. **Cryptography and Network Security - Principles and Practice**. 2. ed.Prentice-Hall, 1999.

[STAMP 2002] STAMP, Mark. **Digital Rights Management: The Technology Behind The Hype**. Technical Paper, USA, August, 2002, 16 p.

[SUN XACML] Sun Microsystems, Inc. **Sun's XACML Implementation**. Disponível em http://sunxacml.sourceforge.net/javadoc/index.html .

[RIVEST 1995] RIVEST R. **The RC5 Encryption Algorithm**. *Dr. Dobb´s Journal*, January 1995.

[RIVEST 1978] RIVEST R. L., SHAMIR A., ADLEMAN L. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**, *Communications of The ACM*, v.21, n.2, p.120-126, February 1978.

[VOLLBRECHT 2000] VOLLBRECHT J. et al. **AAA Authorization Framework**. The Internet Society, RFC-2904, August 2000. Disponível em http://www.ietf.org/rfc/rfc2904.txt .

[WESTPHALL 2000] WESTPHALL Carla M.. **Um Esquema de Autorização para a Segurança de Sistemas Distribuídos de Larga Escala**. Tese de Doutorado, PGEEL-UFSC, Florianópolis, SC, dezembro de 2000.

ANEXO – CÓDIGO DA POLÍTICA XACML

As figuras 44, 45, 46 e 47 apresentam a seqüência do código da política implementada em XACML para a validação do protótipo apresentado no capítulo 7 deste trabalho. A figura 44 apresenta a parte inicial do código da política definindo qual recurso esta política se destina e também quais sujeitos e suas ações.

```
- <Policy PolicyId="GeneratedPolicy"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
    algorithm:ordered-permit-overrides">
   <Description>Essa politica se aplica a todos os usuarios do sistema e a
      qualquer acao ... sobre o conteudo presente em
      www.livropdf.com</Description>
  - <Target>
    - <Subjects>
        <AnySubject />
      </Subjects>
    - <Resources>
      - <Resource>
        - <ResourceMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
            equal">
           <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#a
              nyURI">http://www.livropdf.com/</AttributeValue>
           <ResourceAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:re
              source-id"
              DataType="http://www.w3.org/2001/XMLSchema#a
              nyURI" />
          </ResourceMatch>
        </Resource>
      </Resources>
    - <Actions> <AnyAction /> </Actions> </Target>
```

Figura 44 – Política XACML.(parte 1).

A figura 45 apresenta a primeira regra definida na política , onde é permito a impressão do recurso aos sujeitos classificados como ”A”.

```
- <Rule RuleId="Rule1" Effect="Permit">
  - <Target>
    - <Subjects>
      - <Subject>
        - <SubjectMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:strin
            g-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSche
              ma#string">A</AttributeValue>
            <SubjectAttributeDesignator AttributeId="Classe-Id"
              DataType="http://www.w3.org/2001/XMLSche
              ma#string" />
          </SubjectMatch>
        </Subject>
      </Subjects>
    - <Resources>
        <AnyResource />
      </Resources>
    - <Actions>
      - <Action>
        - <ActionMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:strin
            g-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSche
              ma#string">Print</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:action
              :action-id"
              DataType="http://www.w3.org/2001/XMLSche
              ma#string" />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
</Rule>
```

Figura 45 – Política XACML.(parte 2).

A figura 46 apresenta a segunda regra da política que permite a sujeitos de classe “B” apenas7 visualizar o recurso em questão.

```
<Rule RuleId="Rule2" Effect="Permit">
  - <Target>
    - <Subjects>
      - <Subject>
        - <SubjectMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:strin
            g-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSche
              ma#string">B</AttributeValue>
            <SubjectAttributeDesignator AttributeId="Classe-Id"
              DataType="http://www.w3.org/2001/XMLSche
              ma#string" />
          </SubjectMatch>
        </Subject>
      </Subjects>
    - <Resources>
        <AnyResource />
      </Resources>
    - <Actions>
      - <Action>
        - <ActionMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:strin
            g-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSche
              ma#string">View</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:action
              :action-id"
              DataType="http://www.w3.org/2001/XMLSche
              ma#string" />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
</Rule>
```

Figura 46 – Política XACML.(parte 3).

Finalmente a figura 47 apresenta a regra que permite a aos sujeitos de classe “A” visualizar o recurso.

```
- <Rule RuleId="Rule3" Effect="Permit">
  - <Target>
    - <Subjects>
      - <Subject>
        - <SubjectMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:strin
            g-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSche
              ma#string">A</AttributeValue>
            <SubjectAttributeDesignator AttributeId="Classe-Id"
              DataType="http://www.w3.org/2001/XMLSche
              ma#string" />
          </SubjectMatch>
        </Subject>
      </Subjects>
    - <Resources>
        <AnyResource />
      </Resources>
    - <Actions>
      - <Action>
        - <ActionMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:strin
            g-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSche
              ma#string">View</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:action
              :action-id"
              DataType="http://www.w3.org/2001/XMLSche
              ma#string" />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
  <Rule RuleId="FinalRule" Effect="Deny" />
</Policy>
```

Figura 47 – Política XACML.(parte 4).